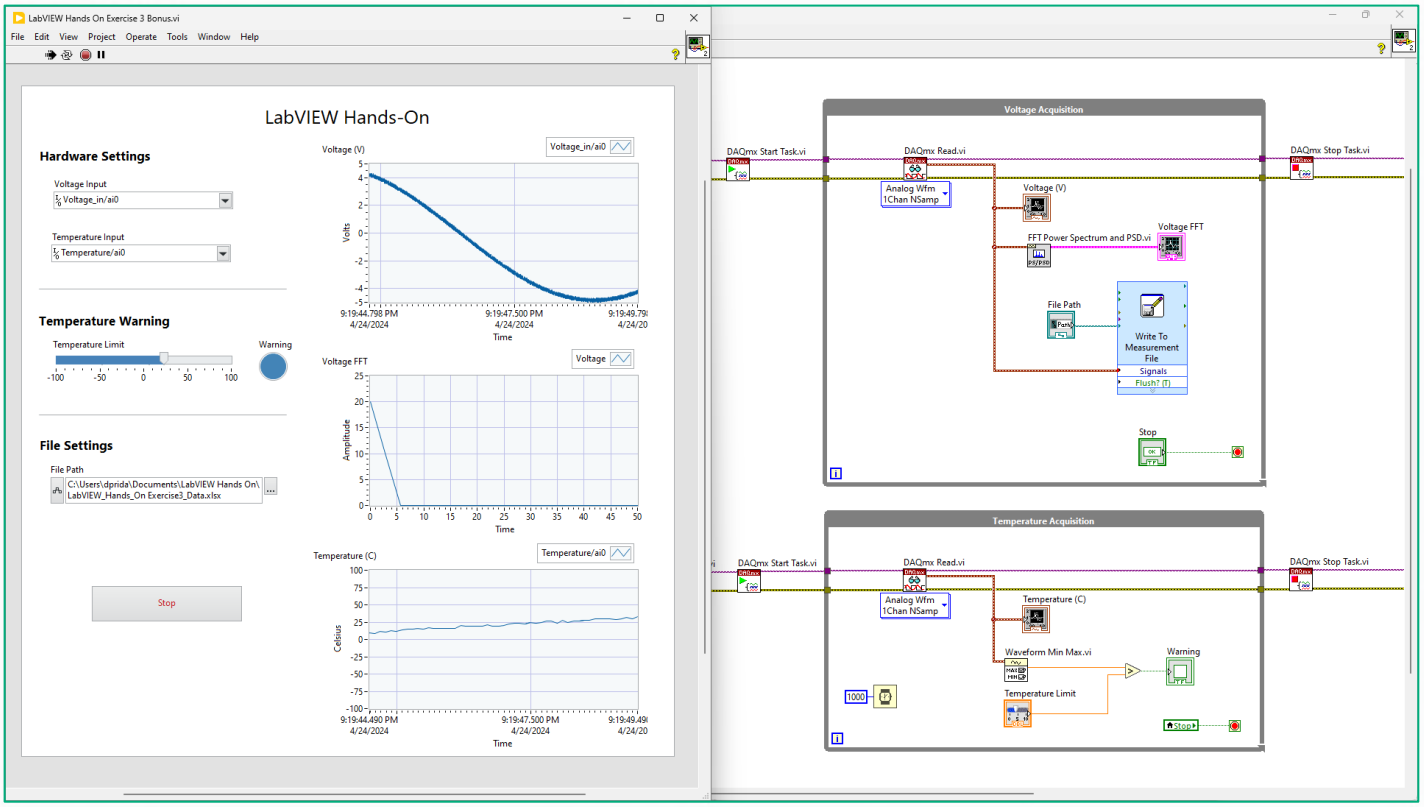
NI is now part of Emerson.

EMERSON



HANDS-ON MANUAL

LabVIEW

Learning How to Use the Most Productive Development Environment for Test and Measurement

ni.com/labview

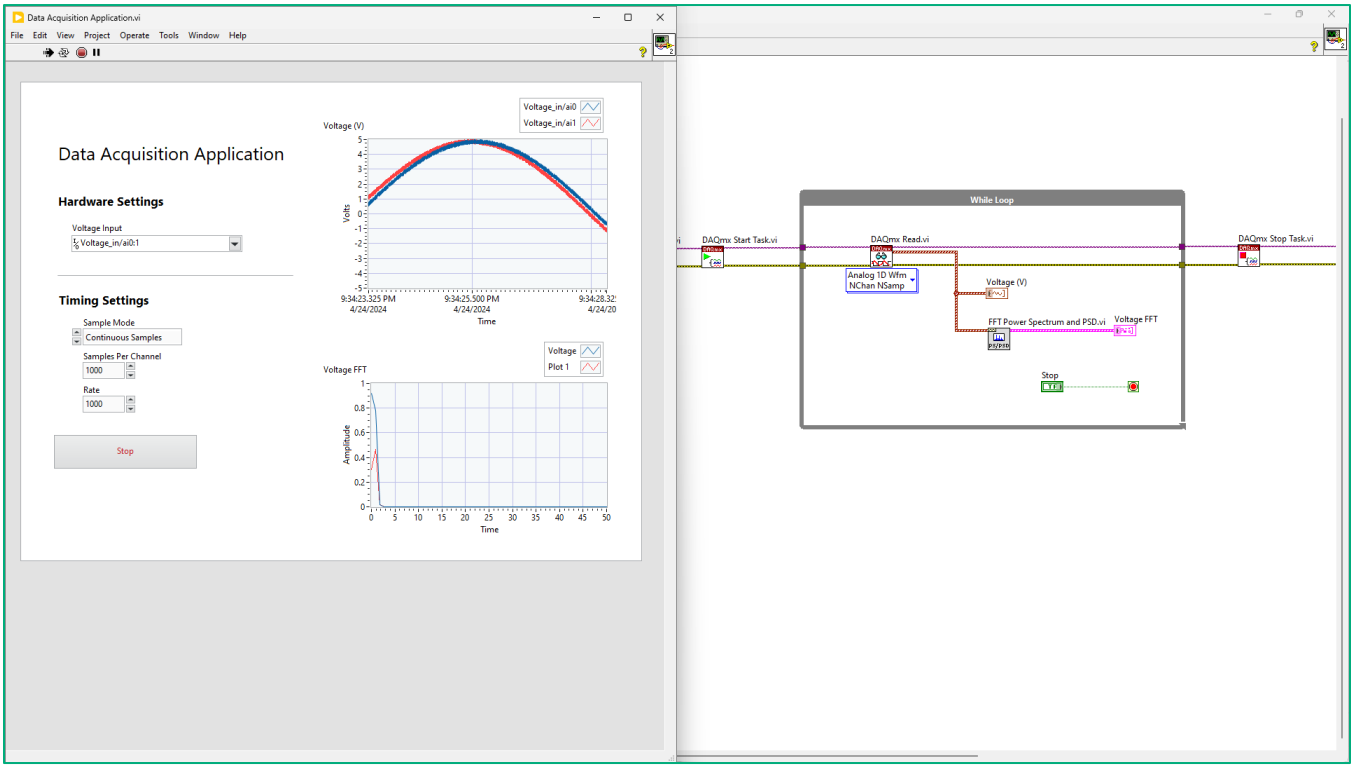
LabVIEW Hands-On Manual Table of Contents

Introduction

Welcome to LabVIEW! These hands-on exercises are designed to help you develop a solid foundation in LabVIEW:

- In Exercise 1, you will learn the basics of programming by building a simple interactive program.
- In Exercise 2, you will turn the simple program into a more practical data acquisition program.
- In Exercise 3, you will expand the data acquisition program to acquire data from multiple devices.
- In Exercise 4, you can view example programs, which are the best references for getting started and building larger applications.

Hardware is not required for these exercises. We will simulate hardware using NI MAX.



NI LabVIEW

LabVIEW is a graphical programming environment that provides unique productivity accelerators for test system development, such as an intuitive approach to programming, connectivity to any instrument, and fully integrated user interfaces.

Pre-Work: Hardware Set-Up and Configuration

Goals:

- Use Measurement & Automation Explorer to create and configure a simulated NI-DAQmx device.

Part A: Introduction to NI MAX

Estimated time: 5 minutes.

If you have not already, please download and install LabVIEW and NI-DAQmx from ni.com.

NI MAX manages all NI hardware and software. This application is installed with most NI software packages. In Part A, you will look at the most used features of NI MAX including **Software** and **Devices and Interfaces**.

1. Open NI MAX by double-clicking the icon on the desktop or navigating to **Start » Search » NI MAX**.

2. After MAX launches, expand **My System » Devices and Interfaces**. If you have hardware connected, you should be able to see it:

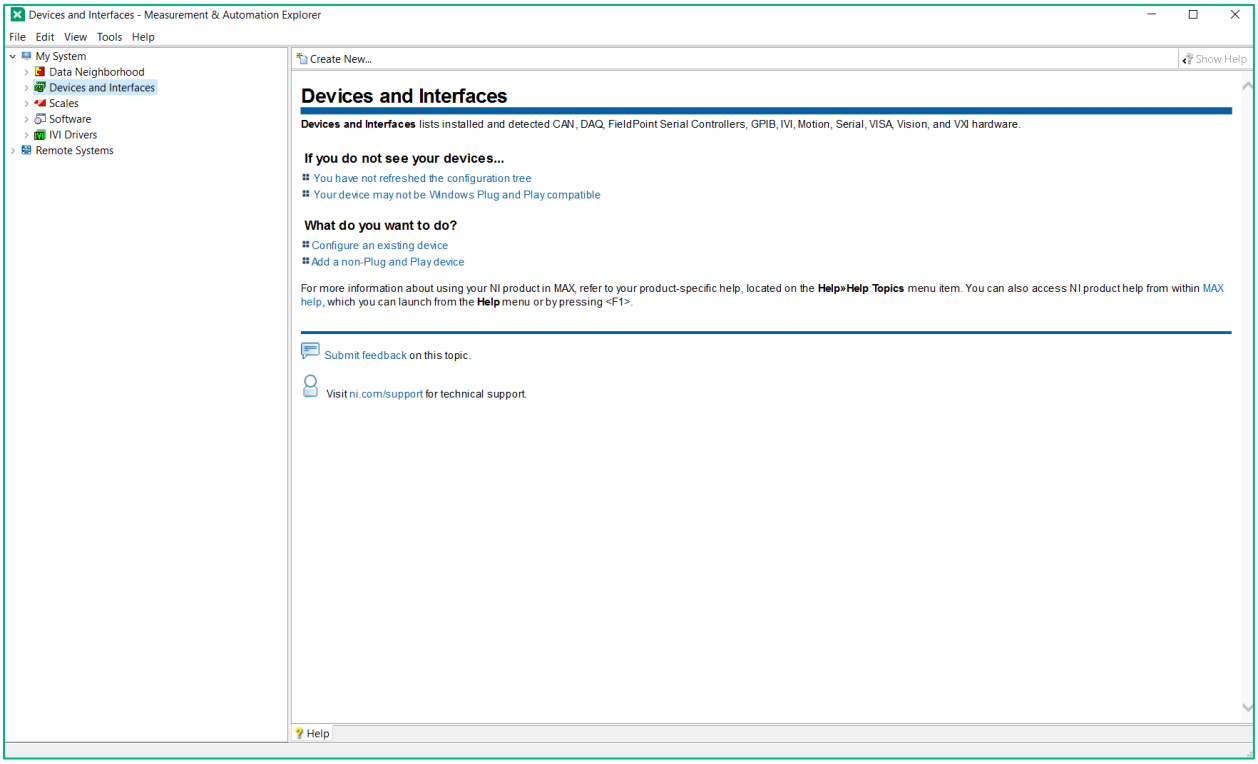


Figure 1. Expand the Devices and Interfaces drop-down menu to show the connected devices.

NI MAX Concept

Within MAX, you can ensure that your hardware is properly connected, check external connections, rename your devices, and even simulate devices for developing code without attached hardware.

3. In addition to the hardware, you can see all the software installed on the machine by expanding **My System » Software**.

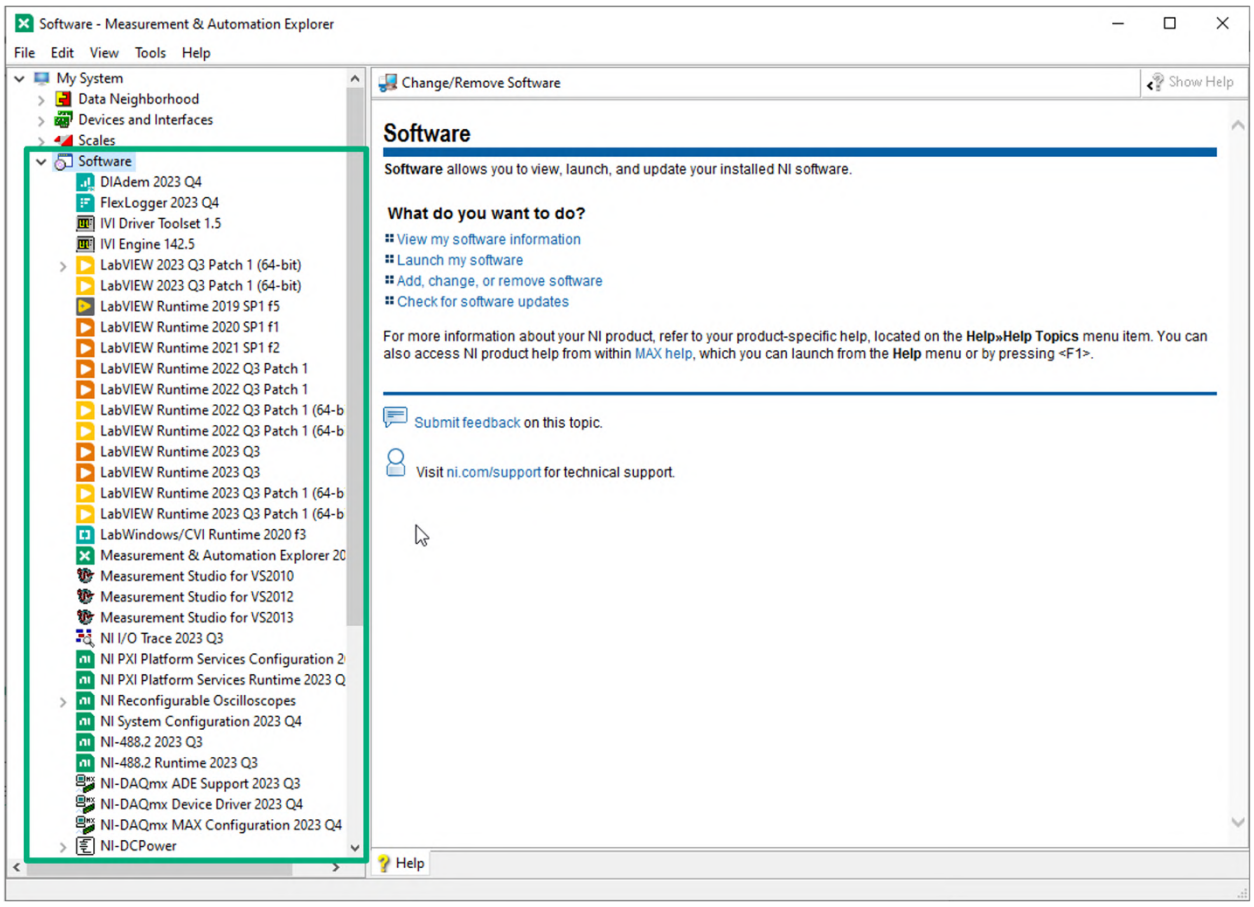


Figure 2. Make sure you have the right software and drivers installed by expanding the Software section.

Exercise Note:

LabVIEW and NI-DAQmx should be listed underneath the software section. NI-DAQmx is the driver that is used for communication with NI data acquisition and signal conditioning devices.

Part B: Creating a Simulated Device

Estimated time: 10 minutes.

You can use MAX to simulate hardware. This enables development and debugging in situations where you do not have access to the actual hardware.

This will also help us create a consistent experience during this guide. The skills you learn will be valuable when programming with your actual, physical hardware - programming for a simulated device and an actual device will be the same.

1. To create a simulated device, right-click on **Devices and Interfaces** and select **Create New…**.

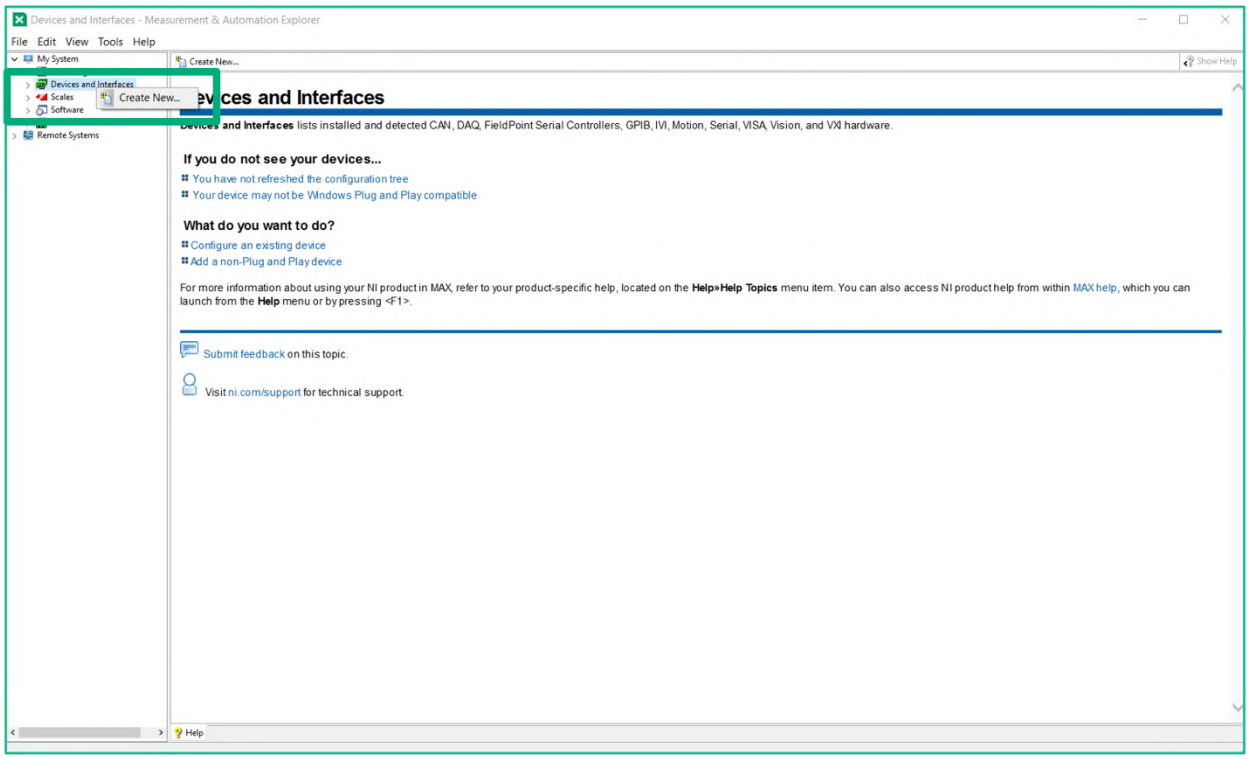


Figure 1. A simulated device enables you to experiment without having the actual hardware on-hand.

2. When the Create New… screen launches, select **Simulated NI-DAQmx Device or Modular Instrument** and press **Finish**.

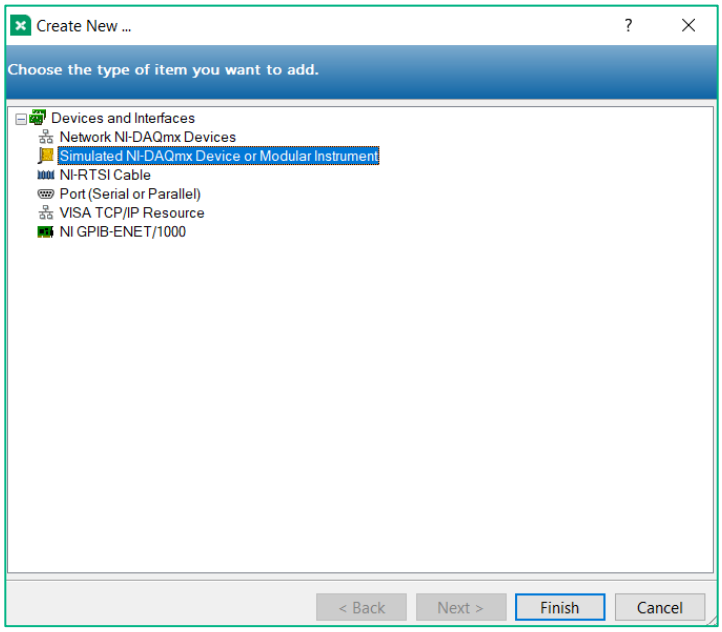


Figure 2. Within MAX, you can simulate any NI-DAQmx device and many modular instruments.

3. In this exercise, we will be simulating a CompactDAQ chassis and various CompactDAQ modules. Here we will add the chassis first. To quickly add the device, type “cDAQ-9178” and select the chassis.

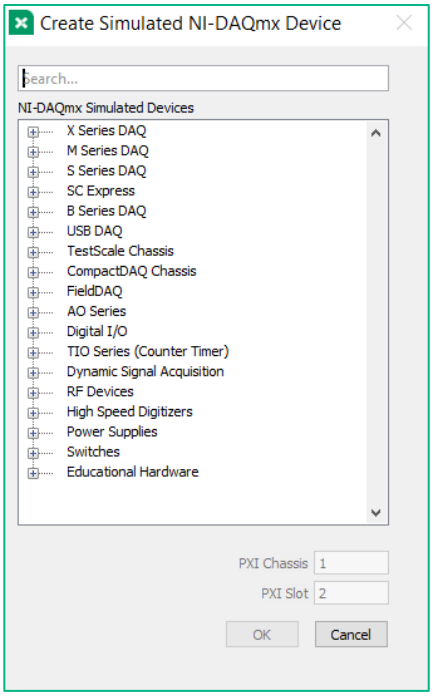


Figure 3. The NI cDAQ-9178 is just one of many options for simulated devices.

4. The new simulated chassis will show up under Devices and Interfaces. Simulated devices are colored yellow. Now that we have simulated our CompactDAQ Chassis we can add our modules to the system. To add modules to our chassis select **Configure Simulated cDAQ Chassis…**

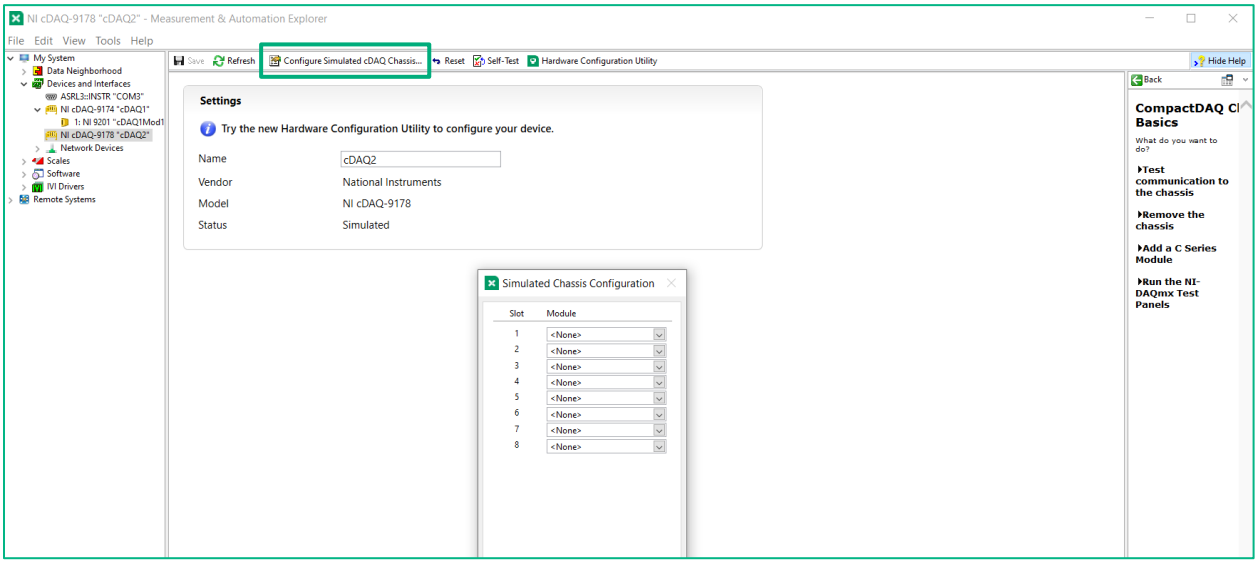


Figure 4. The simulated NI cDAQ-9178 shows up in Devices and Interfaces just like a real device. The emblem of a simulated device will be yellow, while a real device will appear in natural color.

5. Select the modules below (NI 9211, NI 9237, NI 9263, NI 9472, NI 9234, and NI 9215) for the next exercises by going through the list of options shown in the drop down. Press **OK** to add the module to the chassis.

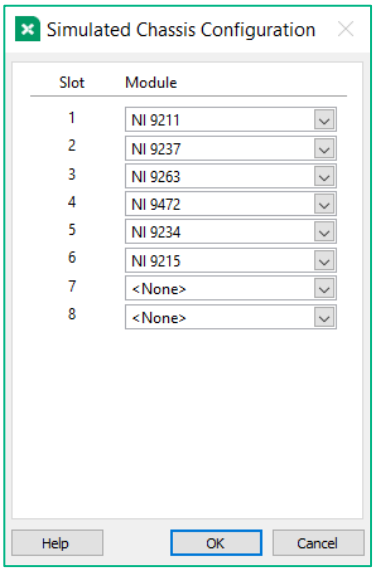


Figure 5. You can configure your simulated chassis for any module that the NI cDAQ-9178 supports.

6. For many applications, simply using the default assigned names is not sufficient. Within MAX, you can rename your modules to match the module type or measurement.

    a. To rename the module, right-click on the module and select **Rename**.

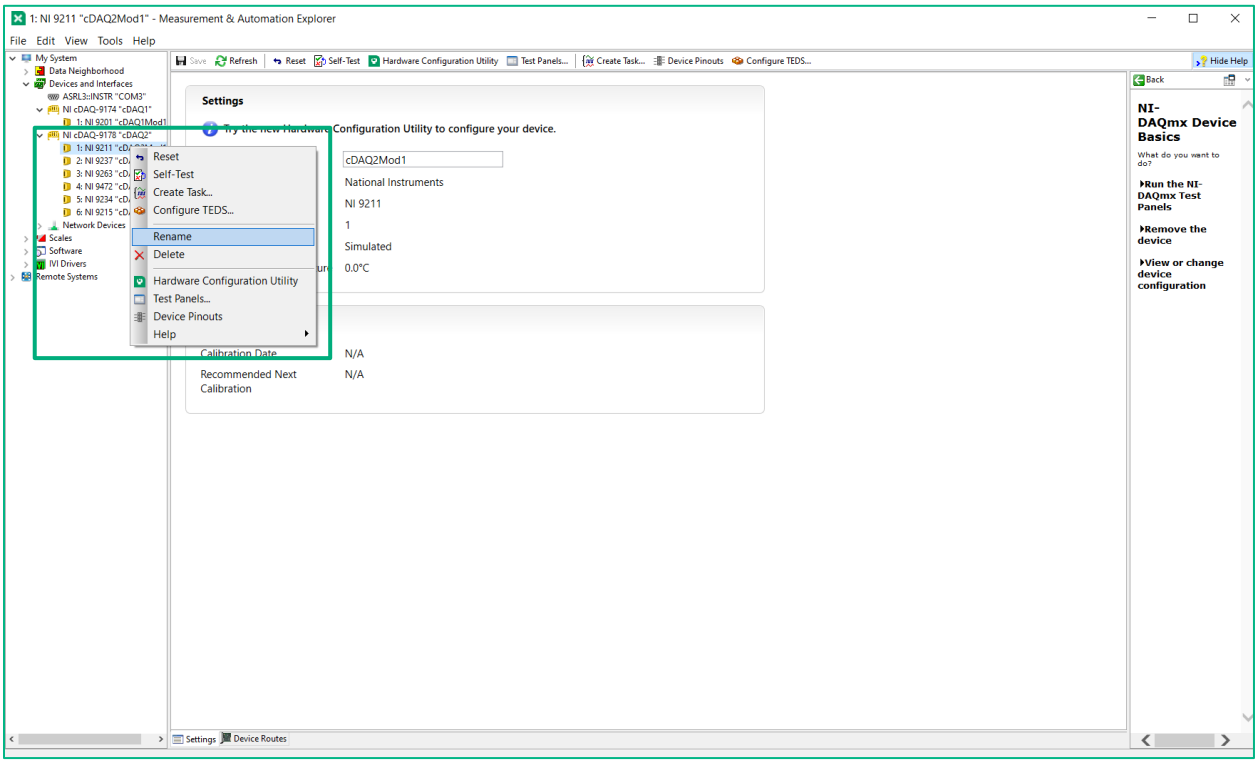
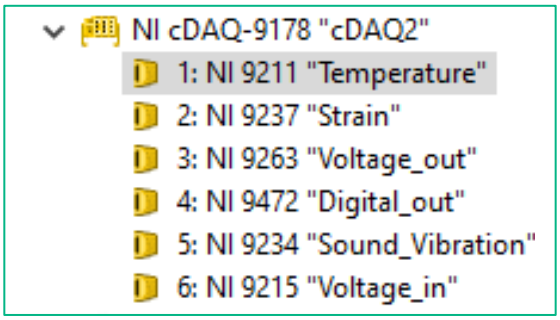


Figure 6. Select Rename **to add new descriptive names.**

    b. Rename each of the channel types to descriptive names:



7. Once the module is added to the simulated chassis, you can perform actions just like on a real device. To illustrate this, right-click on the NI 9211 and select **Test Panels…**

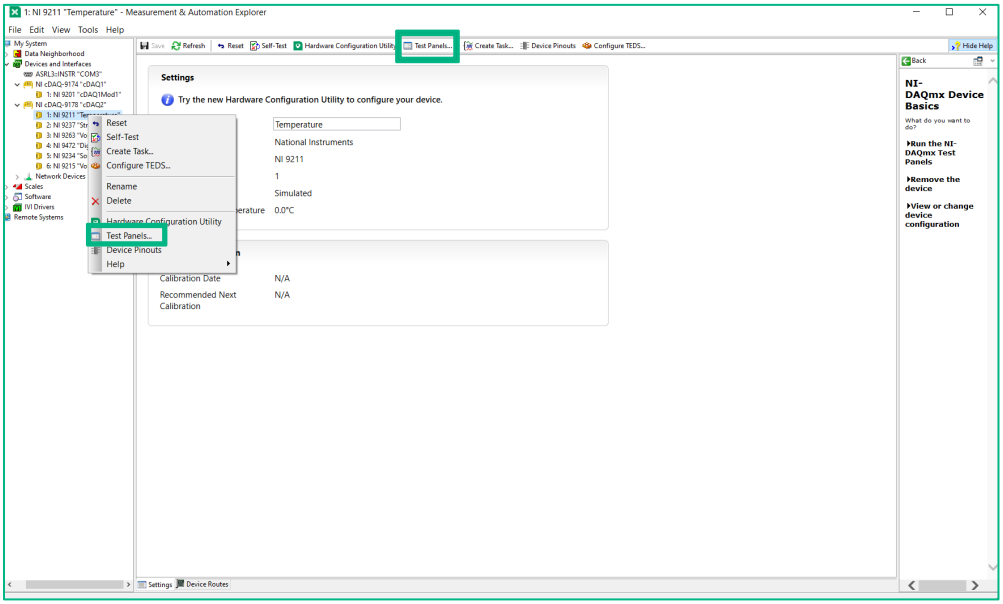


Figure 7. Select Test Panels… **to take measurements from your simulated device.**

8. When the simulated device test panel appears, press **Start** to begin visualizing the simulated data. It is important to note that this data set is built into the NI-DAQmx driver and does not constitute real data; however, you can interact with it in the same way as interacting with real data.

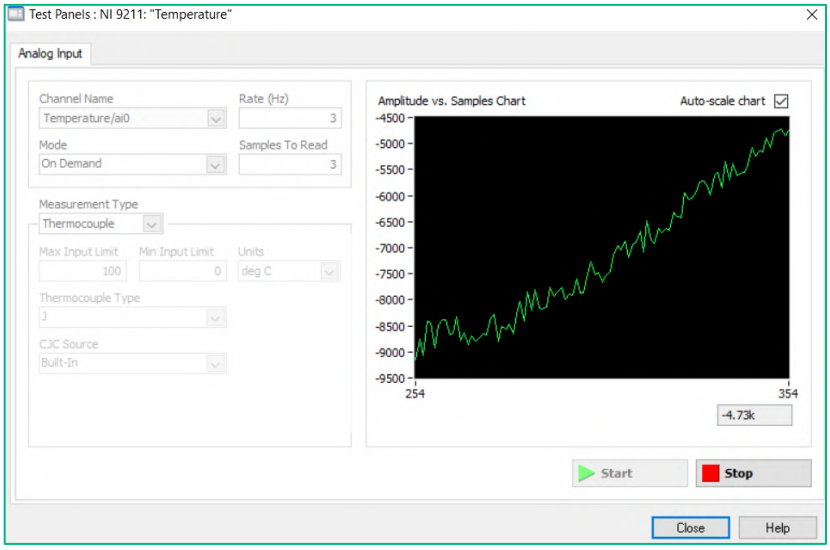


Figure 8. The data displayed in the simulated device test panel is a simple sine wave.

a. Notice with the simulated hardware, you are seeing oscillations in either direction with the temperatures provided. Each of the test panels are slightly different, each providing the inputs for measurements supported by the module (e.g. thermocouple → temperature).

9. Close any open Test Panel windows by selecting **Close**.

10. **Right-click** the chassis under the Devices and Interfaces section and select **Self-Test**.

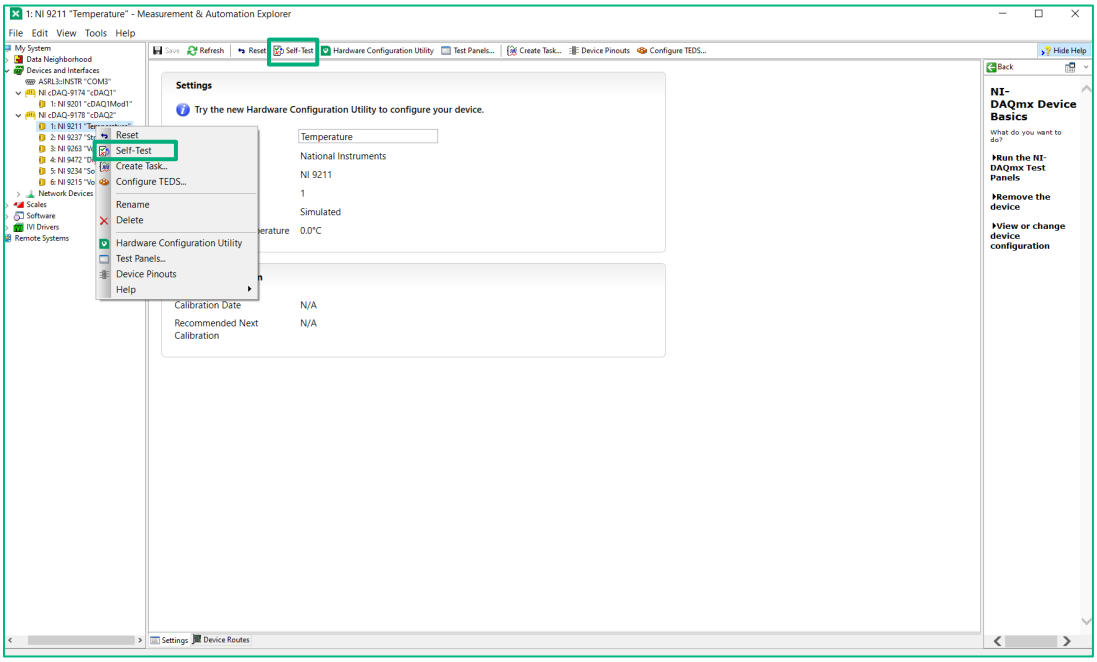


Figure 9. Self-Test is a basic utility for checking connectivity with your hardware.

NI MAX Tip:

Self-testing the chassis is always a good step after simulating or installing hardware.

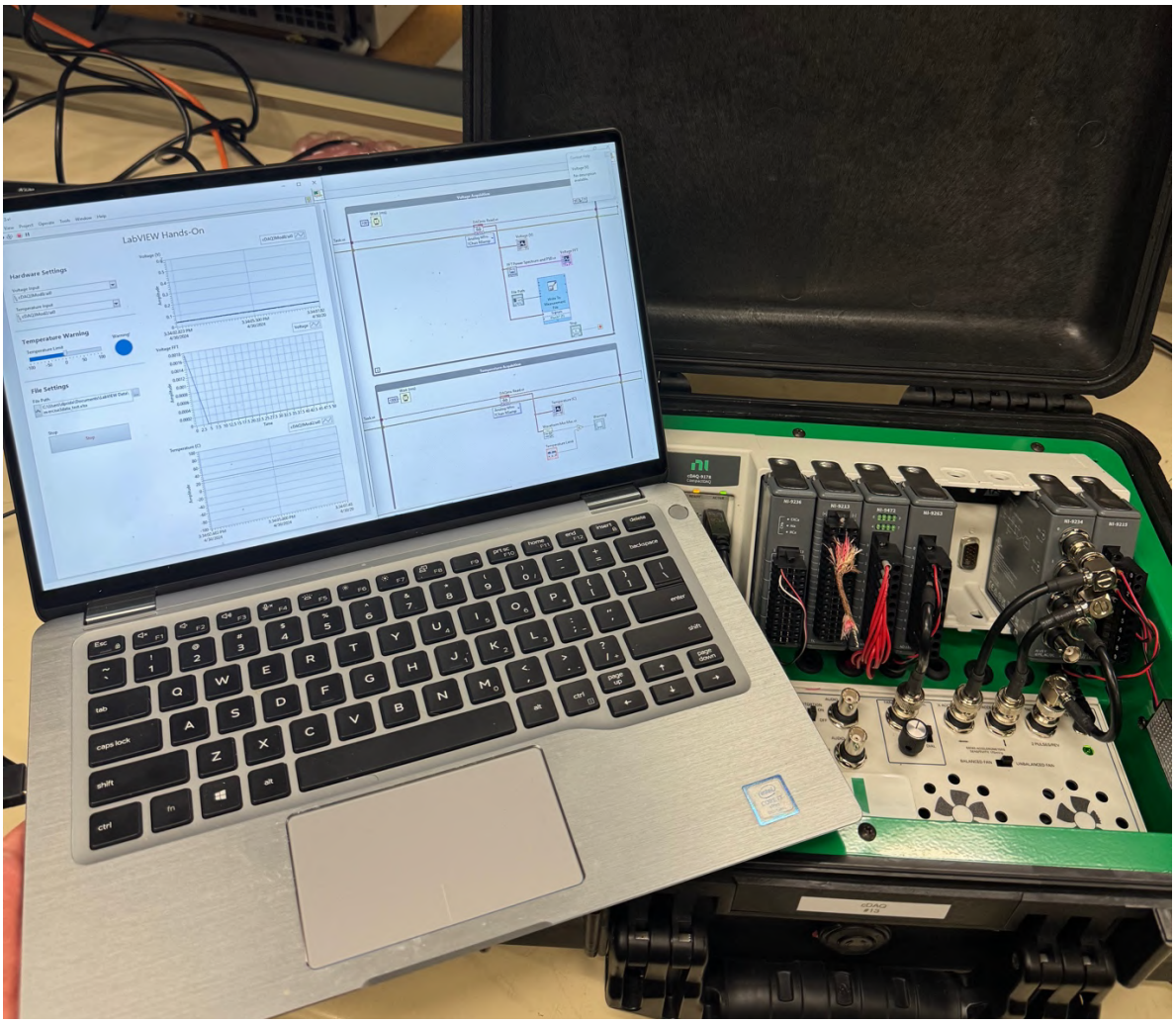
Part C: A Note on Simulated Devices

Estimated time: 5 minutes.

As mentioned, simulated devices are helpful when we do not have access to the actual equipment. The programming and development in LabVIEW for a simulated CompactDAQ device will be **exactly the same** as it would be for an actual CompactDAQ device. The difference will be the data. All NI-DAQmx simulated devices return analog input data in the form of a full-scale sine wave with 3% full-scale noise.

In Part B, a CompactDAQ-9178 with several input and output modules was simulated. And in Exercises 2, 3, and 4, we will learn how to develop a LabVIEW application to acquire data from these simulated devices. I can connect to an actual, physical system, without having to make any changes to my code.

This is the same application from Exercise 3 but connected to an actual CompactDAQ-9178.



<End of Exercise>

Exercise 1: Build Your First Application

Estimated time: 35 minutes.

Goals:

- Learn the basics of LabVIEW: Block Diagram, Front Panel, and Graphical Programming.
- Build your first LabVIEW application.

Part A: Learn the LabVIEW Environment

Estimated time: 5 minutes

A LabVIEW application is called a VI, short for virtual instrument, because it emulates the appearance and operation of physical instruments. There are always two components to a LabVIEW VI:

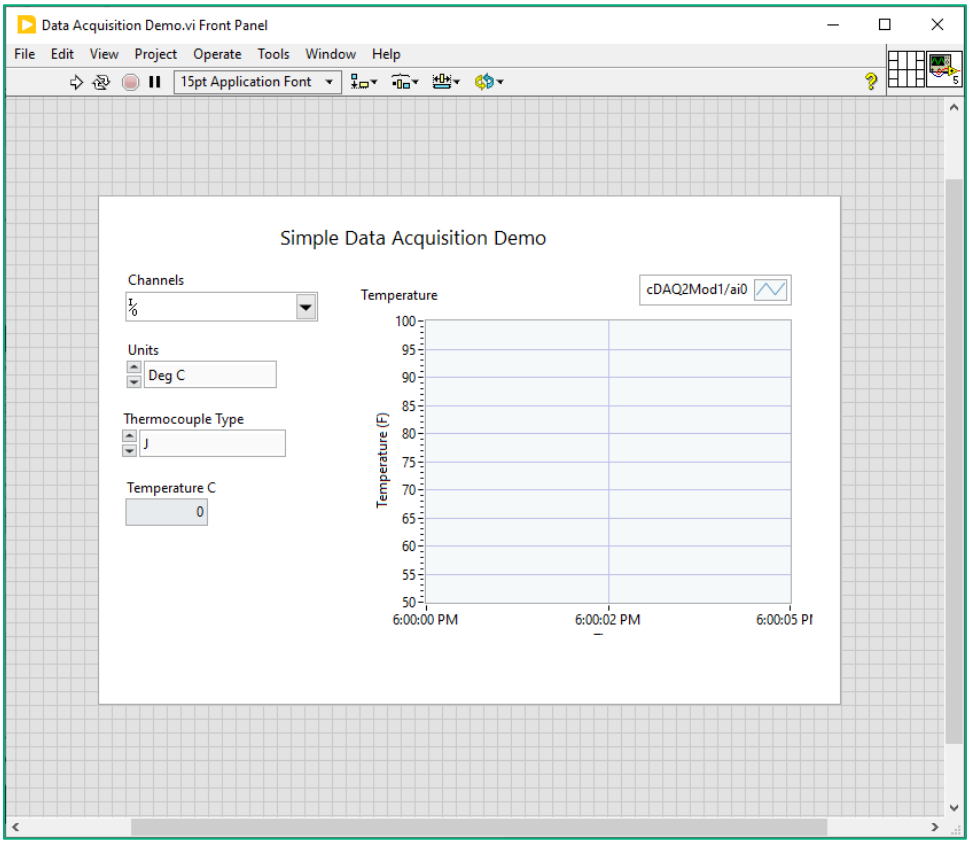


Figure 1. Front Panel of a LabVIEW temperature measurement VI with a chart and temperature display and controls for selecting the device, units, and thermocouple type.

The Front Panel is the user interface for your program. The Front Panel has controls and indicators, which are the interactive input and output terminals, respectively. Controls and indicators placed on the Front Panel are automatically placed on the Block Diagram, too.

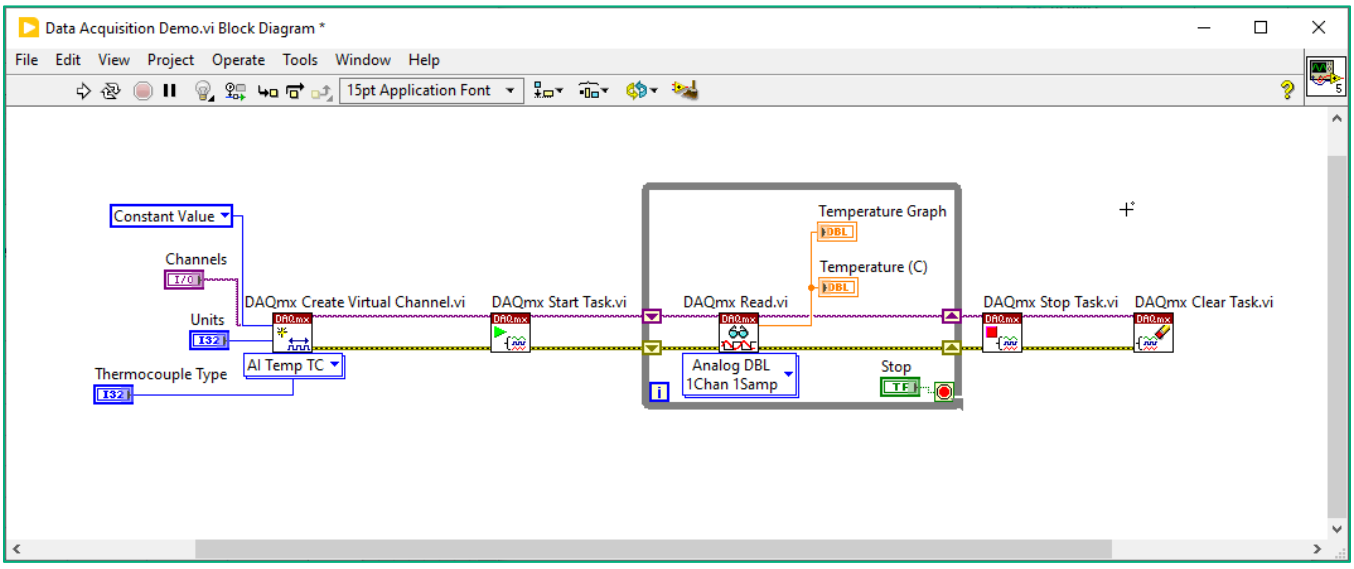


Figure 2. The Block Diagram of the same temperature measurement VI. This uses the DAQmx driver to create a temperature measurement task, read data, and stop and clear the task. On the left, you can see the inputs from the Front Panel. We will learn more about this throughout this guide.

The Block Diagram contains the graphical source code of your LabVIEW program. Block diagram objects include terminals, subVIs, functions, constants, structures, and wires, which transfer data among other block diagram objects.

Graphical Programming

LabVIEW uses graphical programming. Unlike text-based languages, you won't develop your application by writing lines of code. Instead, you'll develop by dragging, dropping, and connecting items onto the Front Panel and Block Diagram. Right-click on the Front Panel or Block Diagram to open the palette and add items.

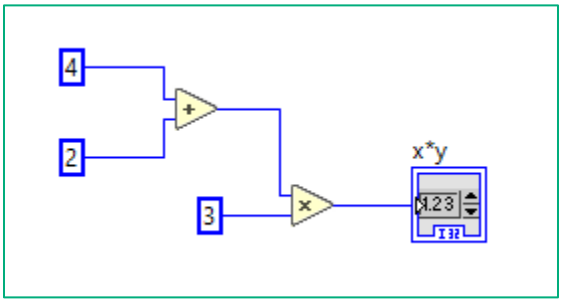


Figure 3. This shows a very simple calculation developed on the Block Diagram. We have three constants, two math functions, addition and multiplication, and an indicator to display the result on the Front Panel. Wires connect it all. The indicator would display 18 as a result.

LabVIEW follows a dataflow model for running VIs. A block diagram node executes when it receives all required inputs. When a node executes, it produces output data and passes the data to the next node in the dataflow path. The movement of data through the nodes determines the execution order of the VIs and functions on the block diagram. In Figure 3, the multiplication function cannot execute until the addition function occurs.

Part B: Build the User Interface

Estimated time: 10 min.

In Part B, we will build the user interface of our program with the Front Panel. You will create a simple user interface with elements to control and monitor a signal.

1. Create a new VI by selecting **File >> New VI.**

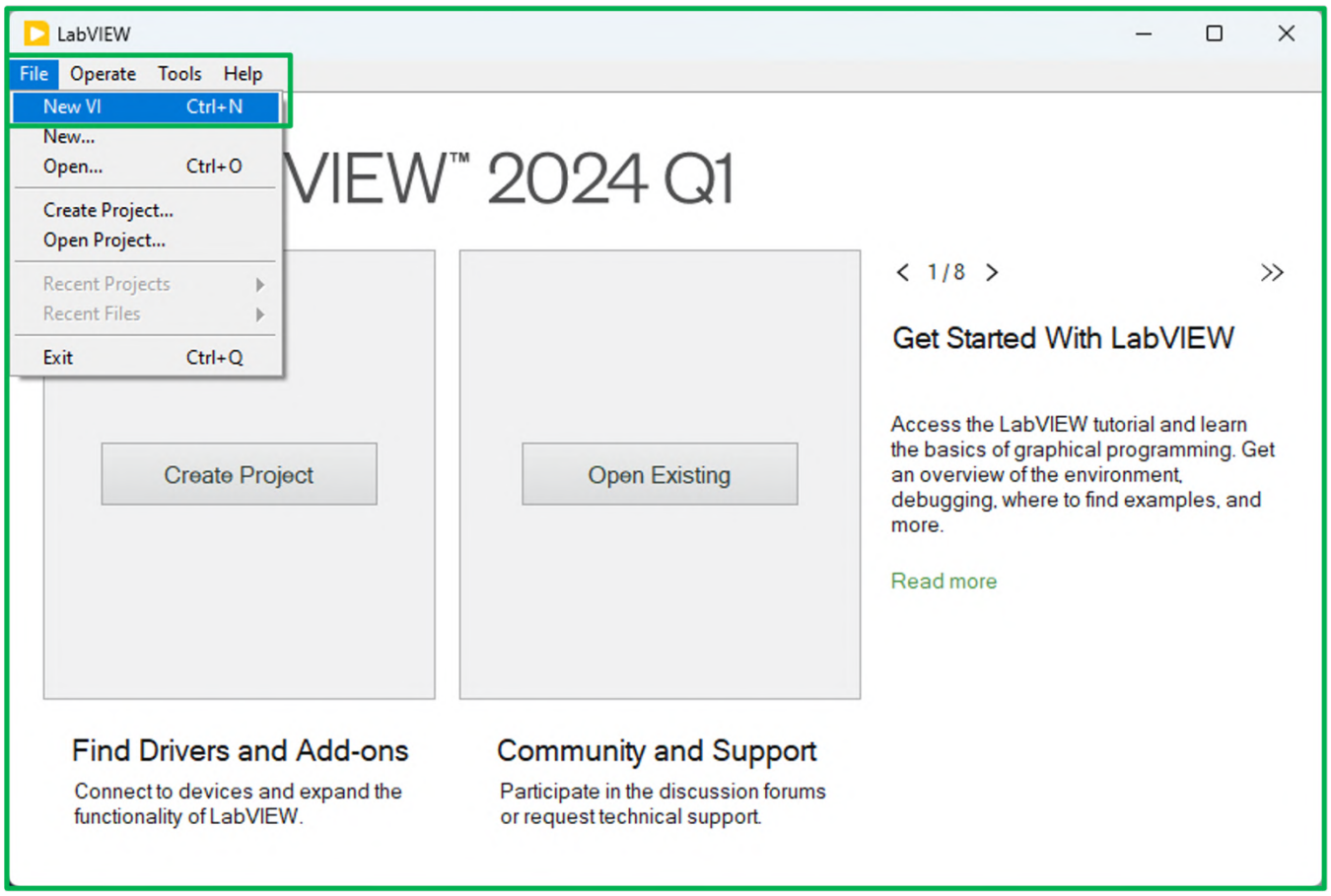


Figure 1. This step creates a new VI, which is made of the Front Panel that serves as the user interface, and the Block Diagram that contains the program code. Every VI will have both.

2. We have organized our screen so the Front Panel is on the left and the Block Diagram is on the right. Right-click the **Front Panel** to view the **Controls Palette**. This contains the various items that you can place on the Front Panel. It is organized by theme, Modern, Silver, Fuse Design System.

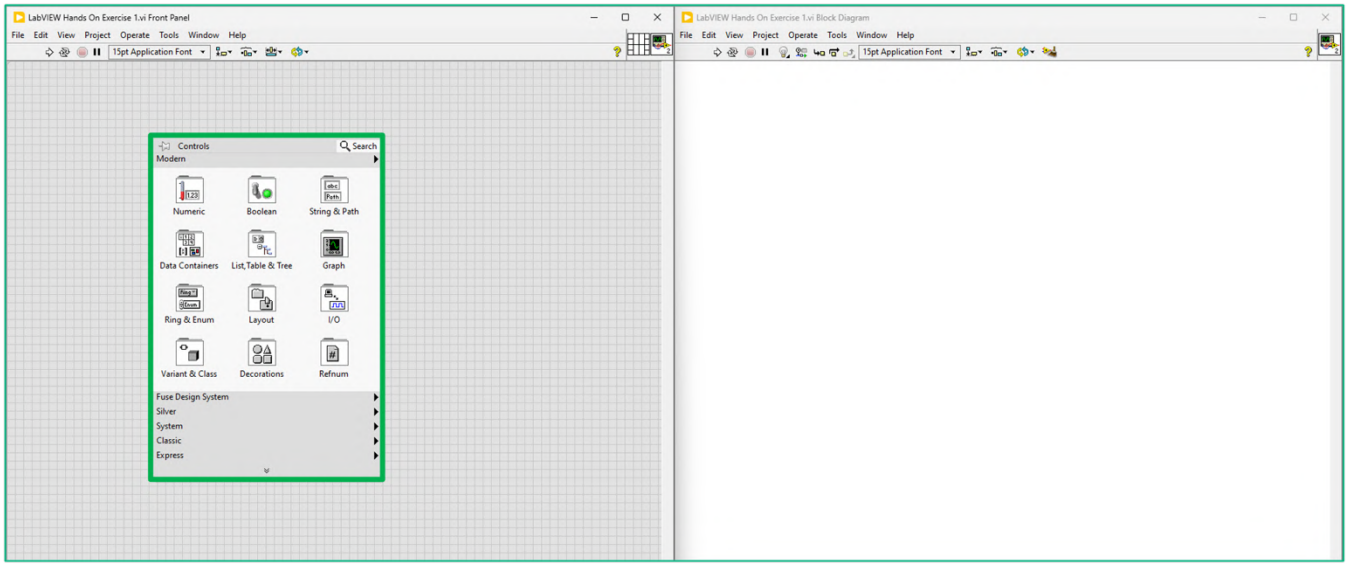


Figure 2. The Controls Palette includes numerous styles of front panel controls and indicators.

3. Click on **Numeric**. You will notice numeric-based objects, like sliders, gauges, dials, and more. Explore the other palettes to find charts, graphs, buttons, and knobs that you can place, as well.

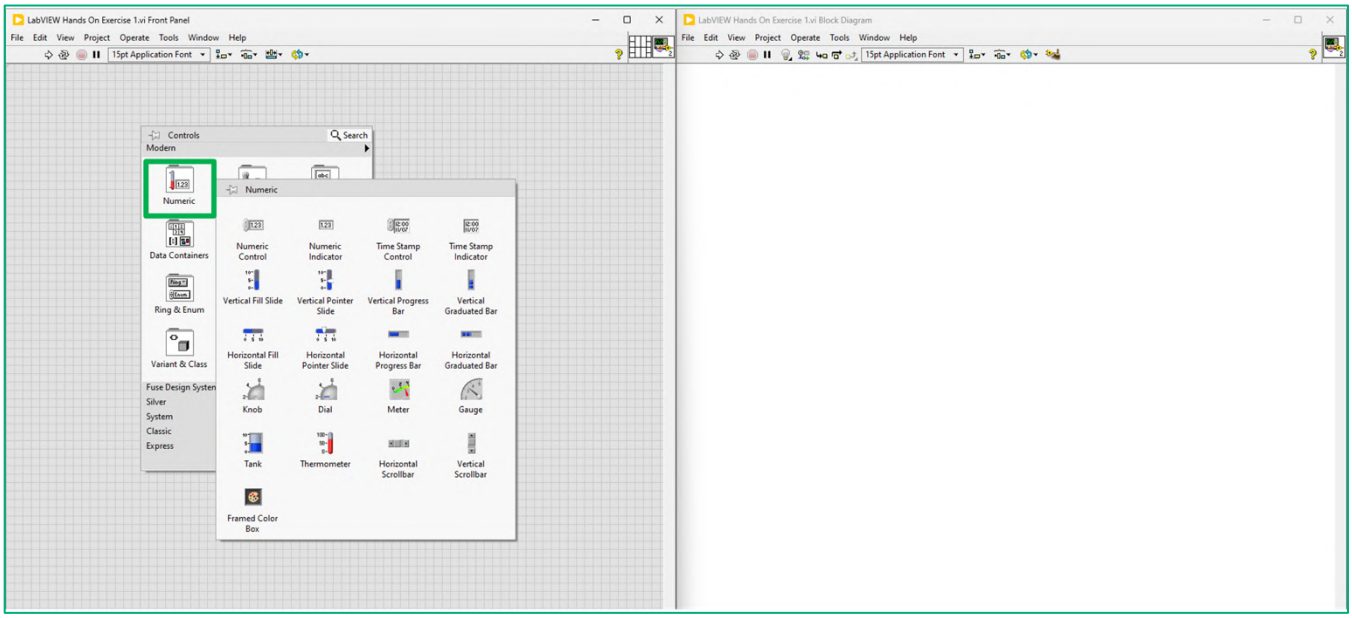


Figure 3. The Controls Palette appears when you right-click the Front Panel and includes all the elements to build your user interface. Take some time to explore the different controls and indicators available.

4. Right-click to open the Controls Palette. Go to the **Fuse Design System** theme, select **Numeric** and click on **Knob**. After you click, you will see an outline of the object.

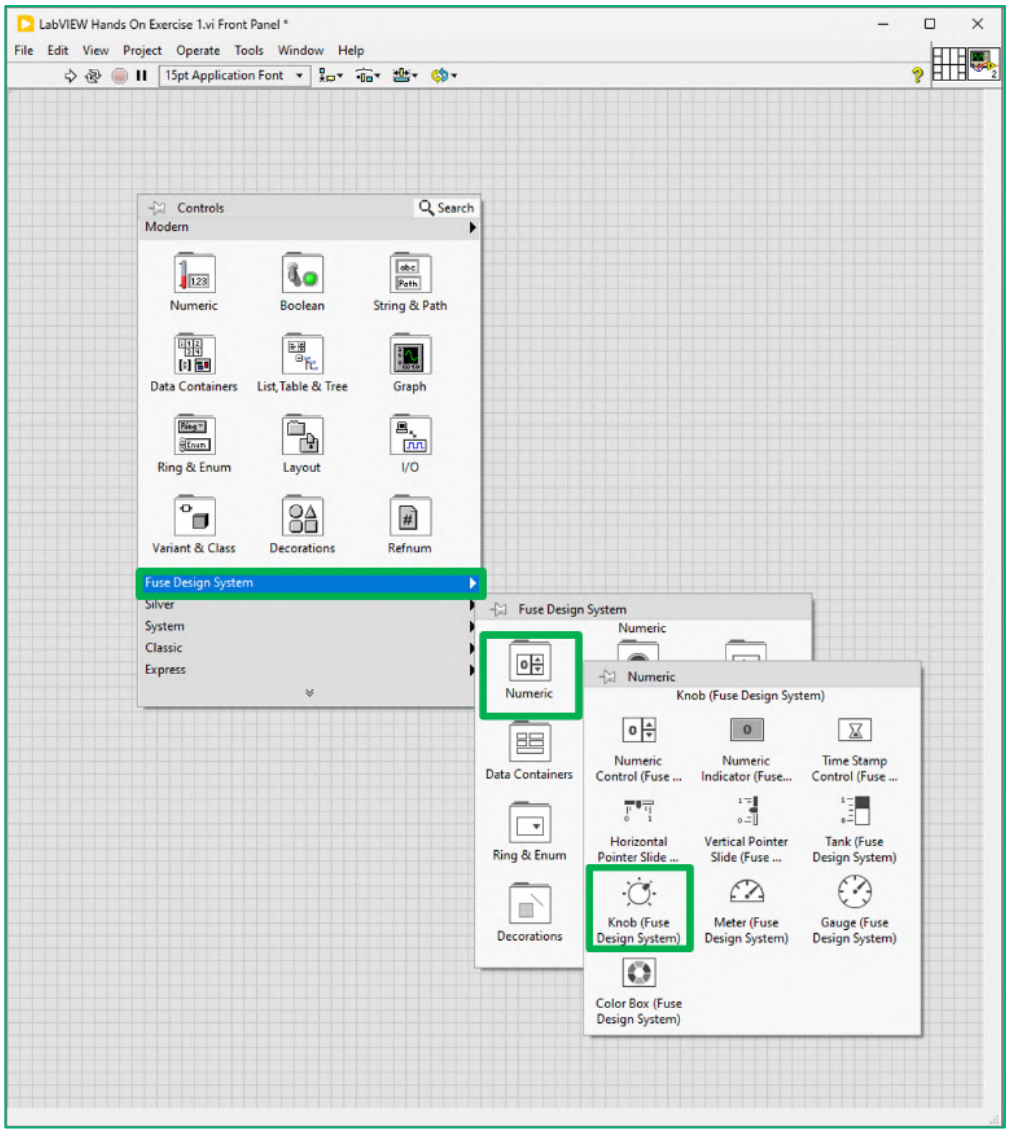


Figure 4. Add a Knob to the Front Panel.

5. **Click** anywhere on the Front Panel to place the item. Try placing this towards the top left of the Front Panel.

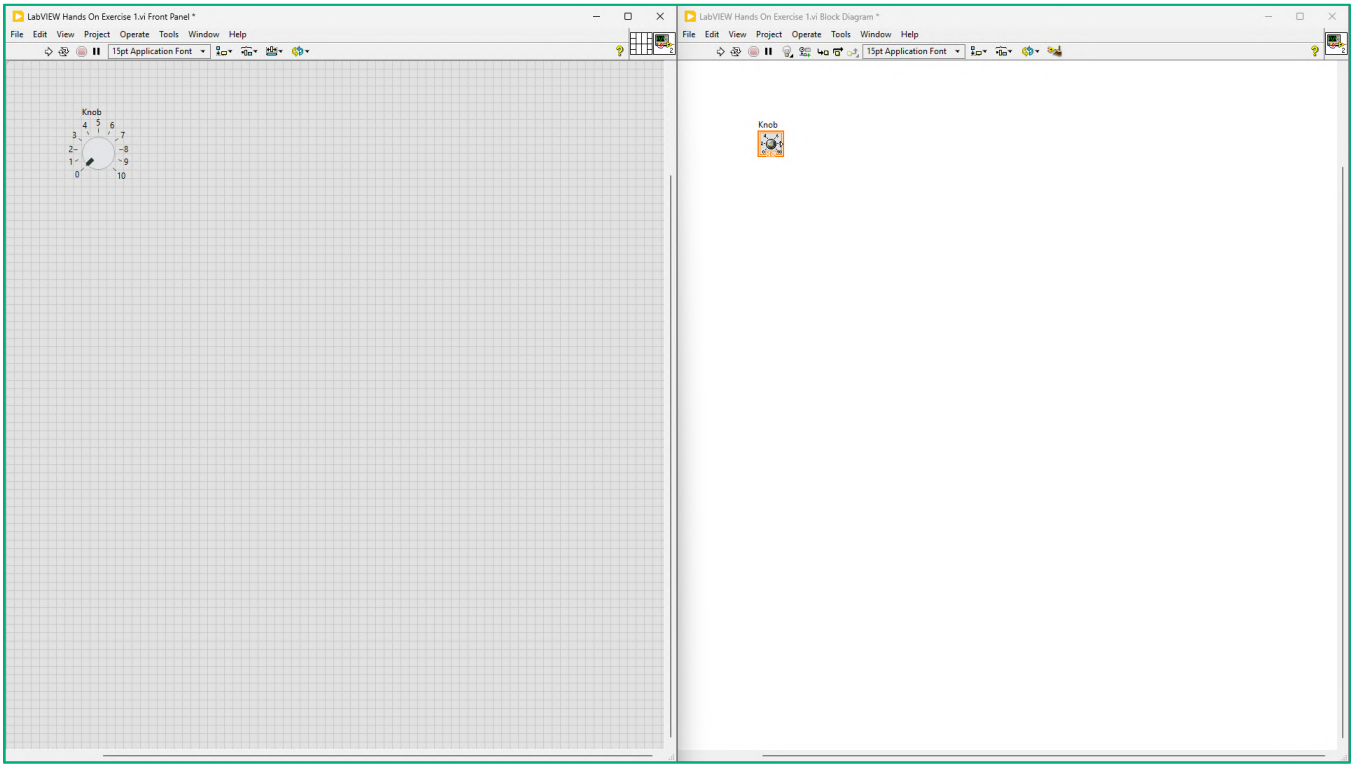


Figure 5. You have placed a Knob on the Front Panel. On the Block Diagram, a Knob element has also been placed. Controls and indicators placed on one are automatically added to the other.

6. Resize the **Knob** using the blue handles that appear when you hover over it or click on it.

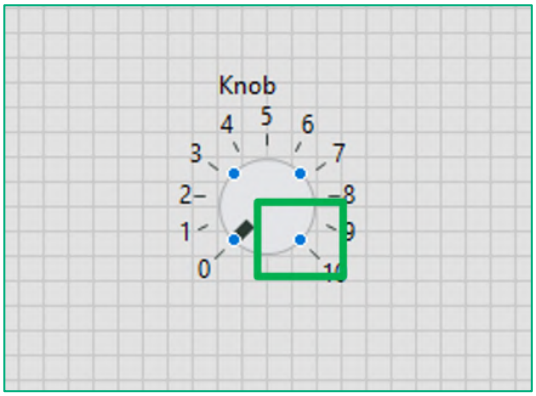


Figure 6. You can move and resize controls and indicators when your cursor is on the object. Resizing handles appear at the points from which you can resize. Use these handles to resize the control to your liking.

7. Right-click the **Knob** control and select **Properties**. Once the Properties window appears, go to the **Scale** tab. Change the Scale **Range Maximum** to **50** and click **OK**.

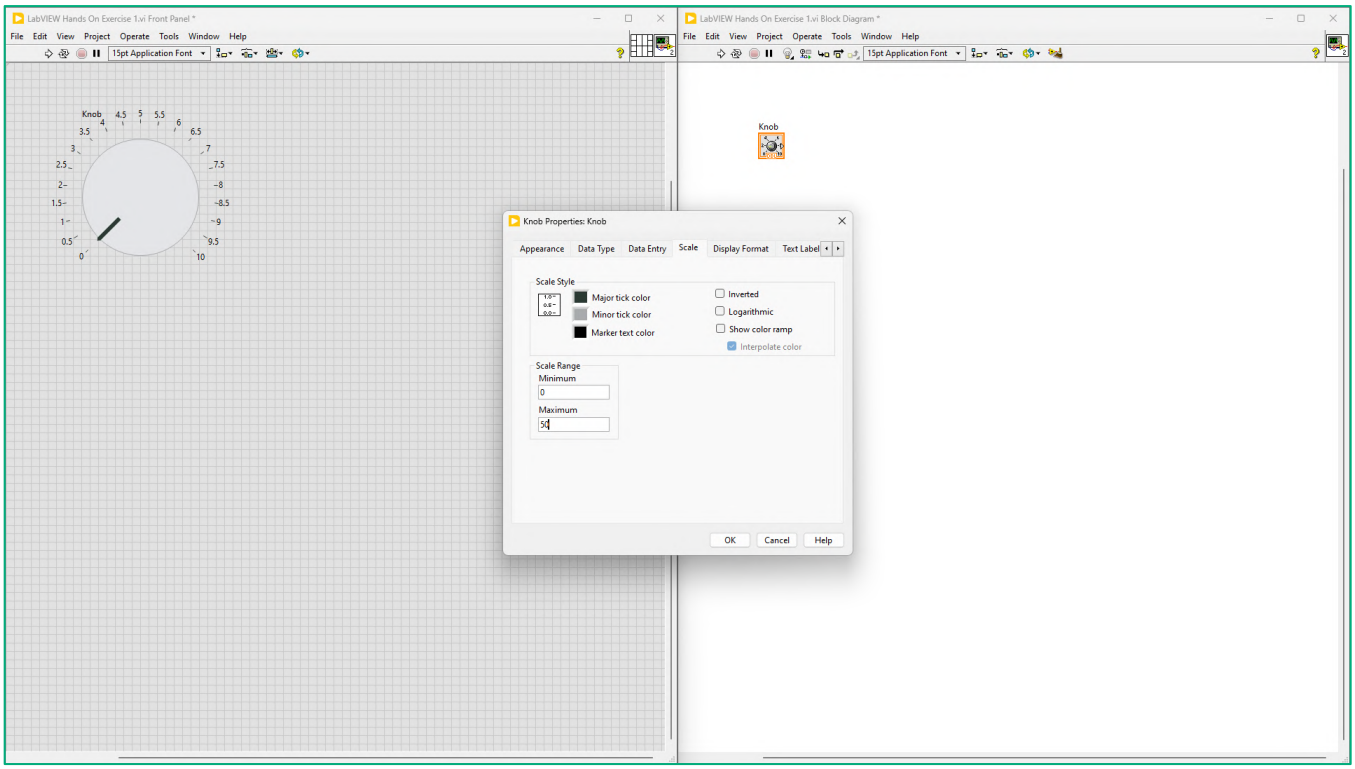


Figure 7. Take some time to explore the different properties you can modify.

LabVIEW Tip:

You can change the maximum range of controls or indicators by double-clicking on the last number on the element. If you double-click on 50, you will be able to change it again without entering the Properties window.

8. Right-click the **Knob** and select **Visible Items >> Digital Display**.

9. Move the **Digital Display** below the knob. To do this, hover over the object, avoid the resizing handles, click and hold on the bottom right corner. Drag the item to the desired location.

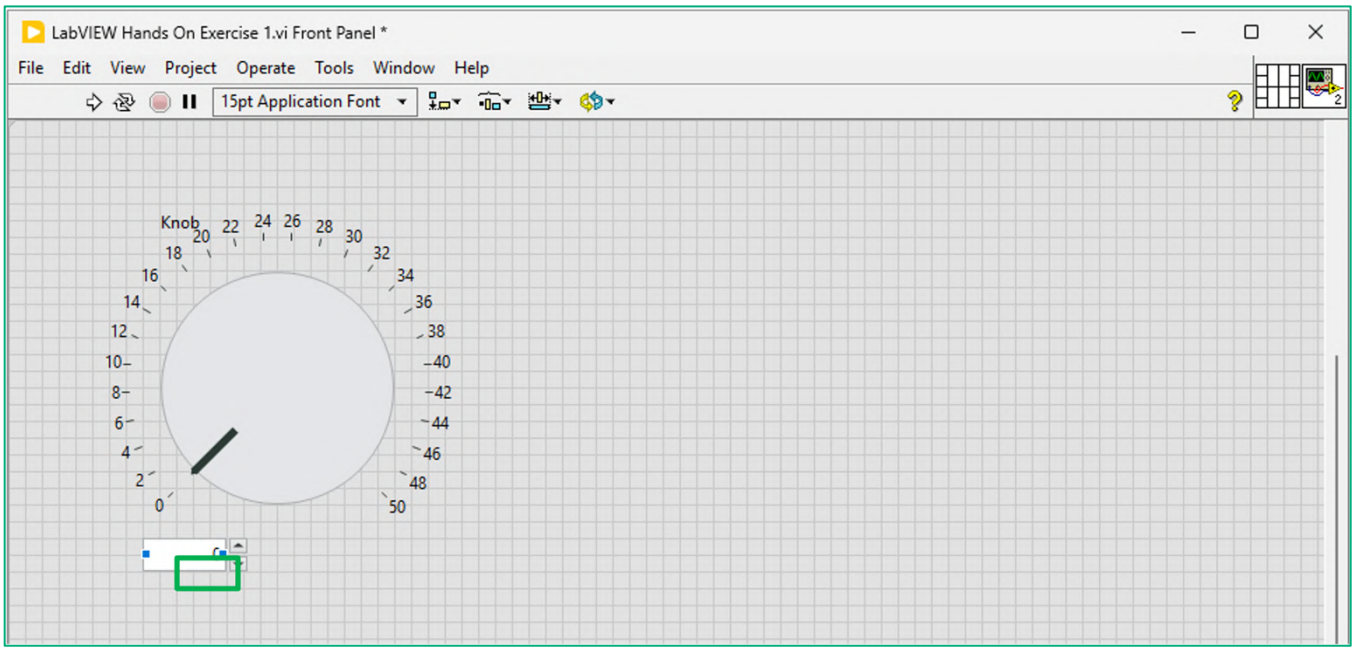


Figure 8. To move an object on the Front Panel. Click and drag the item to the desired location.

10. Double-click the label **"Knob"** and type the new label **"Frequency"**. This will control the frequency of a generated signal.

11. Right-click to open the Controls Palette. Go to the **Fuse Design System >> Graph** and click on **Waveform Graph. Click** to place this item on the Front Panel. This will be used to display the generated signal.

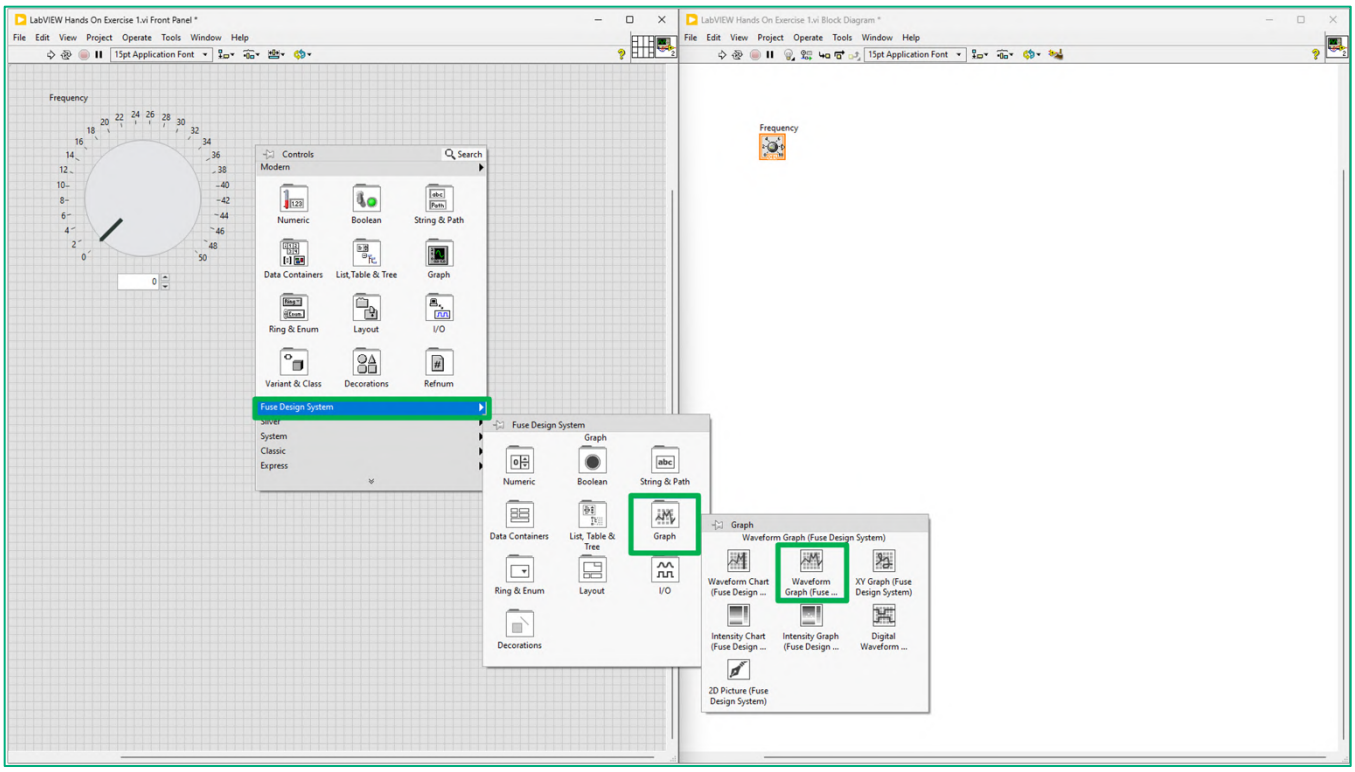


Figure 9. Add a Waveform Graph **to the Front Panel.**

12. Double-click the label **"Waveform Graph"** and type the new label **"Time-Domain Signal"**.

13. Resize the **Time-Domain Signal** graph as desired.

14. Right-click to open the Controls Palette. Go to the **Fuse Design System >> Boolean** and click on **Stop Button.** This will be used to stop the execution of our program. **Click** to place the item.

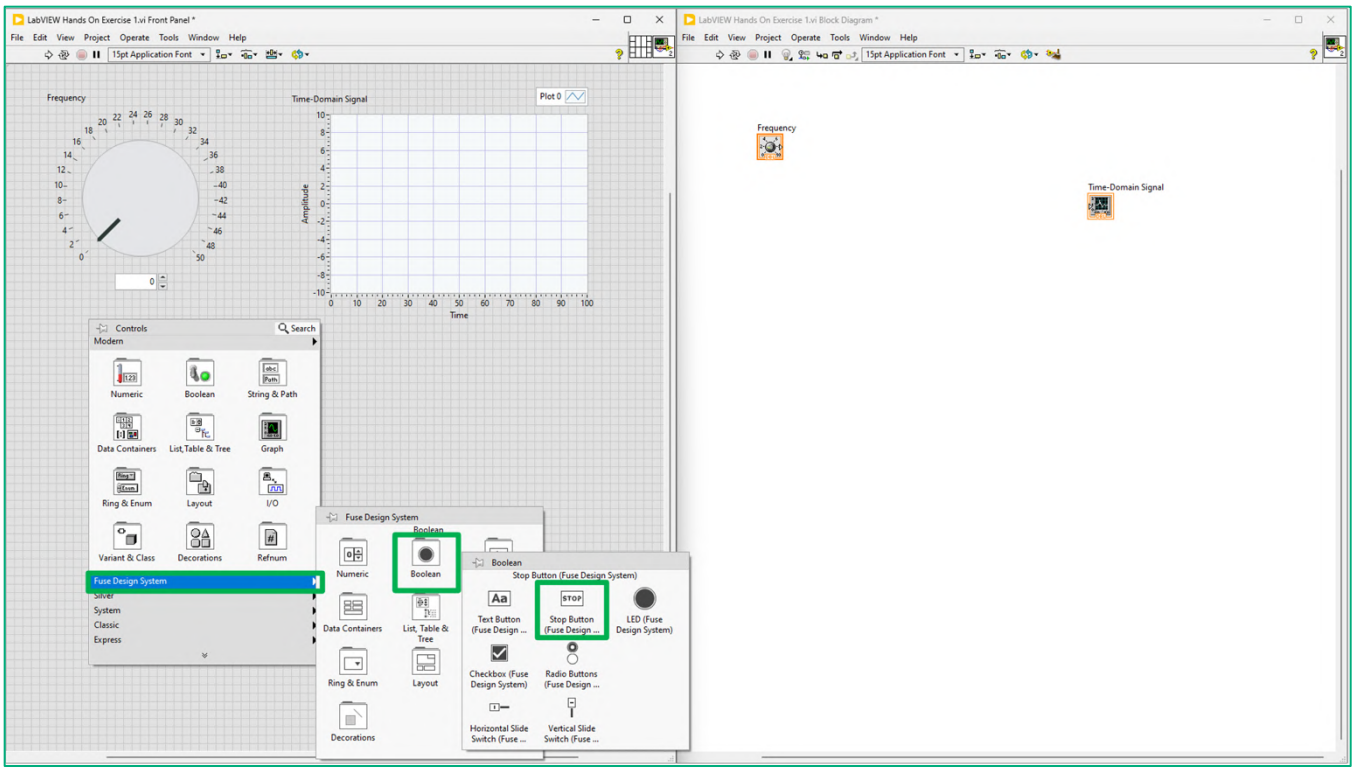


Figure 10. Add a Stop Button **to the Front Panel.**

15. Resize the **Stop** button as desired.

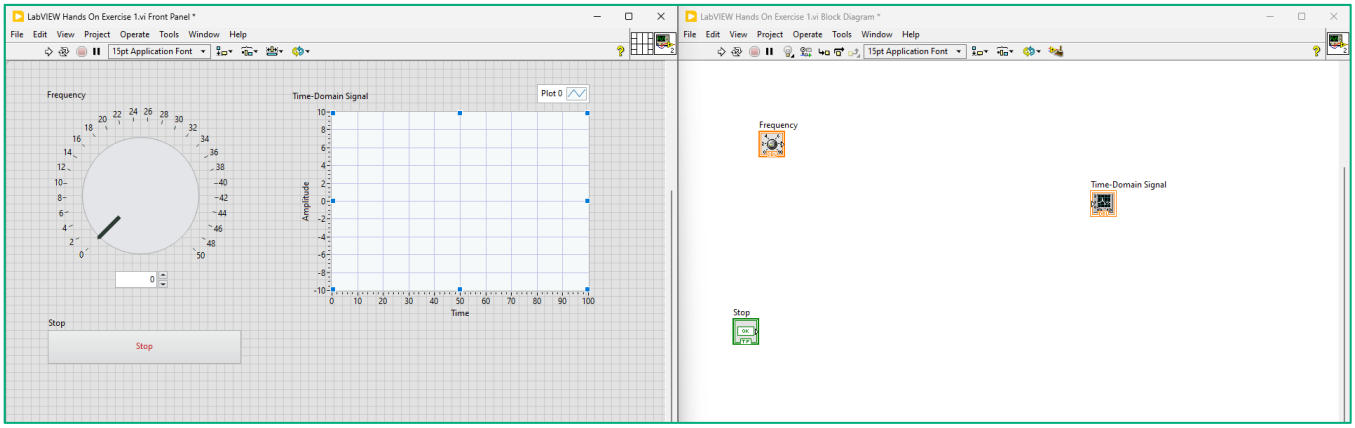


Figure 11. At this point, you will have a knob, a waveform graph, and a stop button on the Front Panel. They are on the Block Diagram, too.

Part C: Building the Block Diagram

Estimated time: 10 min.

In Part C, we will build the functional back end of our application using the Block Diagram. Currently, the Block Diagram has three terminals corresponding to the knob, graph, and stop button we placed in Part B.

1. Right-click the **Block Diagram** and view the **Functions Palette.**

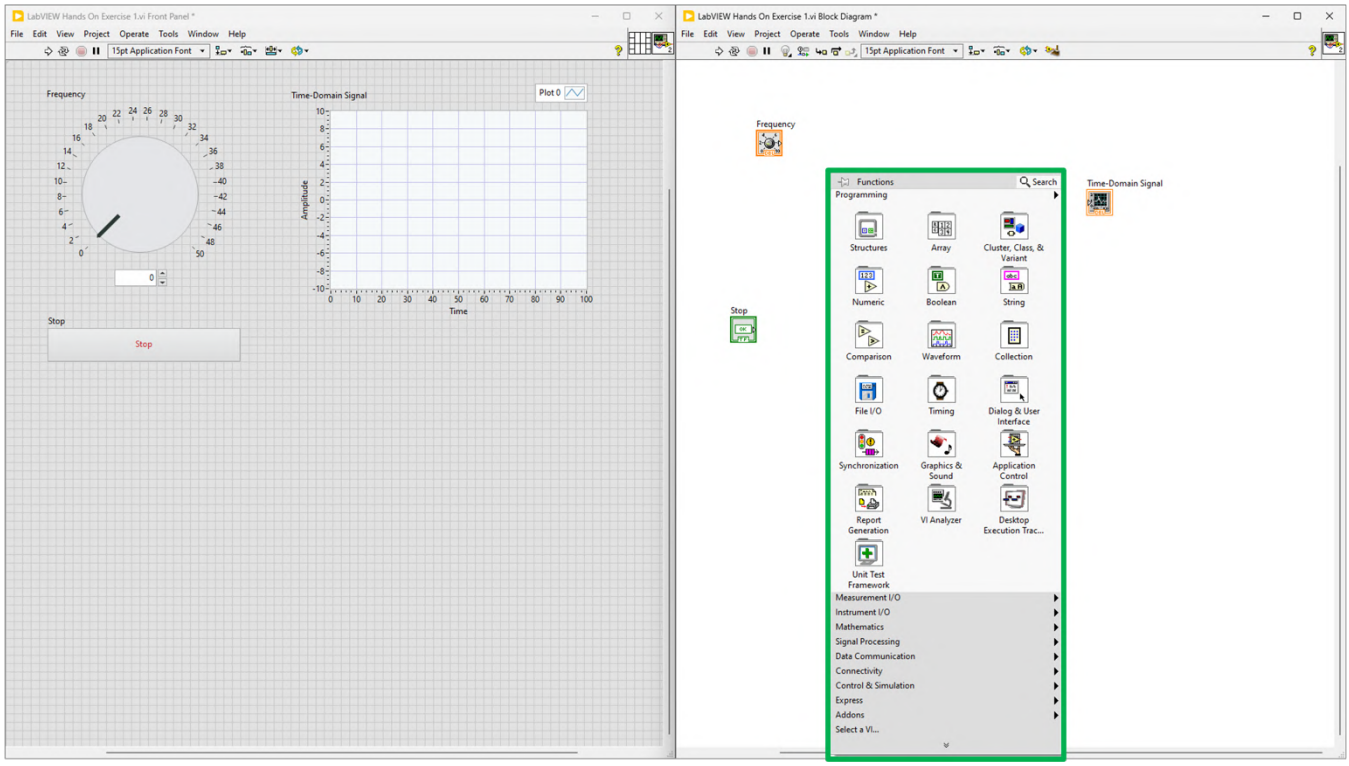


Figure 1. When you right-click the Block Diagram, the Functions palette appears. Here you will find all the necessary elements and functions to create the logic of your program. Take some time to explore the different functions available.

2. In the Functions Palette. Go to **Signal Processing >> Wfm Generation** and click on **Basic Function**. **Click** anywhere on the Block Diagram to place this object.

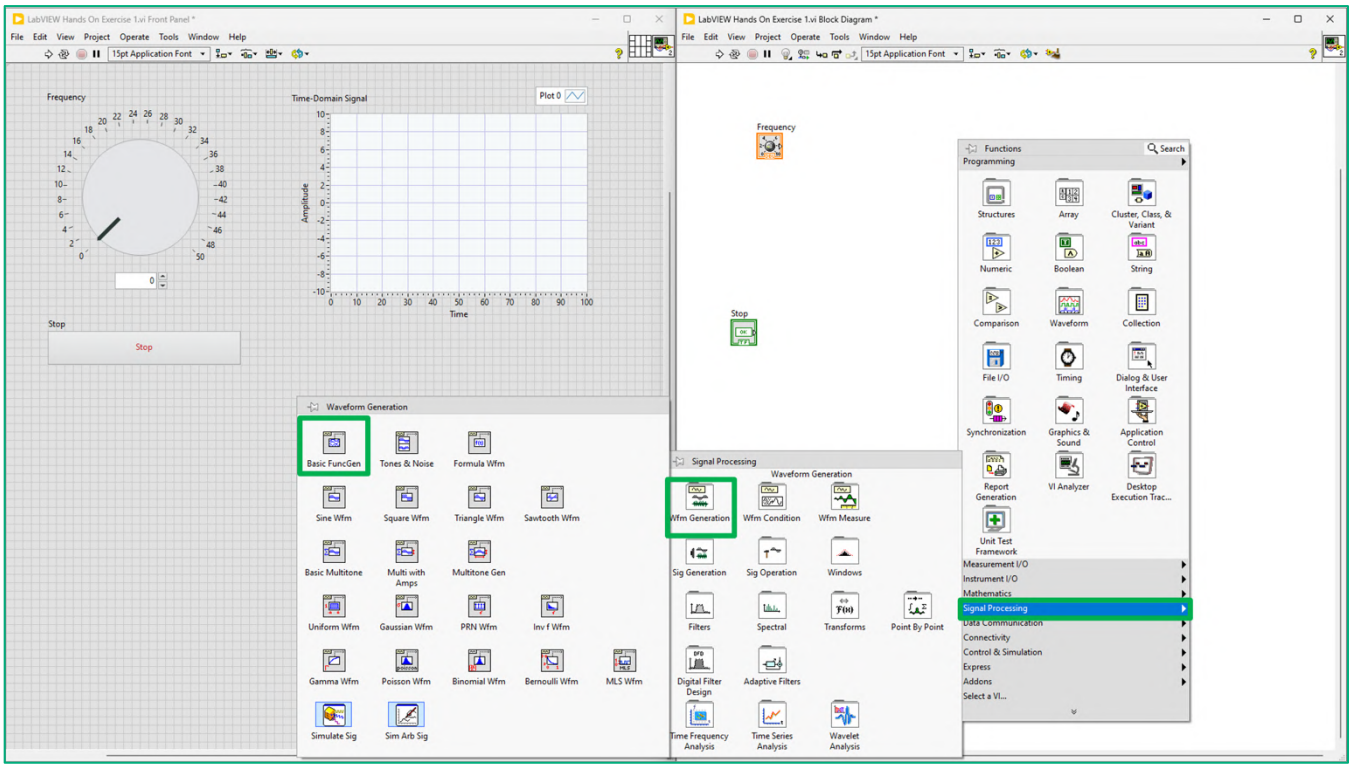


Figure 2. Add a Basic Waveform Function Generator **to the Block Diagram**

3. Type **Ctrl+H** to activate **Context Help.** With Context Help active, hover your cursor over different objects on the Block Diagram and Front Panel. As you do so, the Context Help Window provides details including descriptions and wiring diagrams.

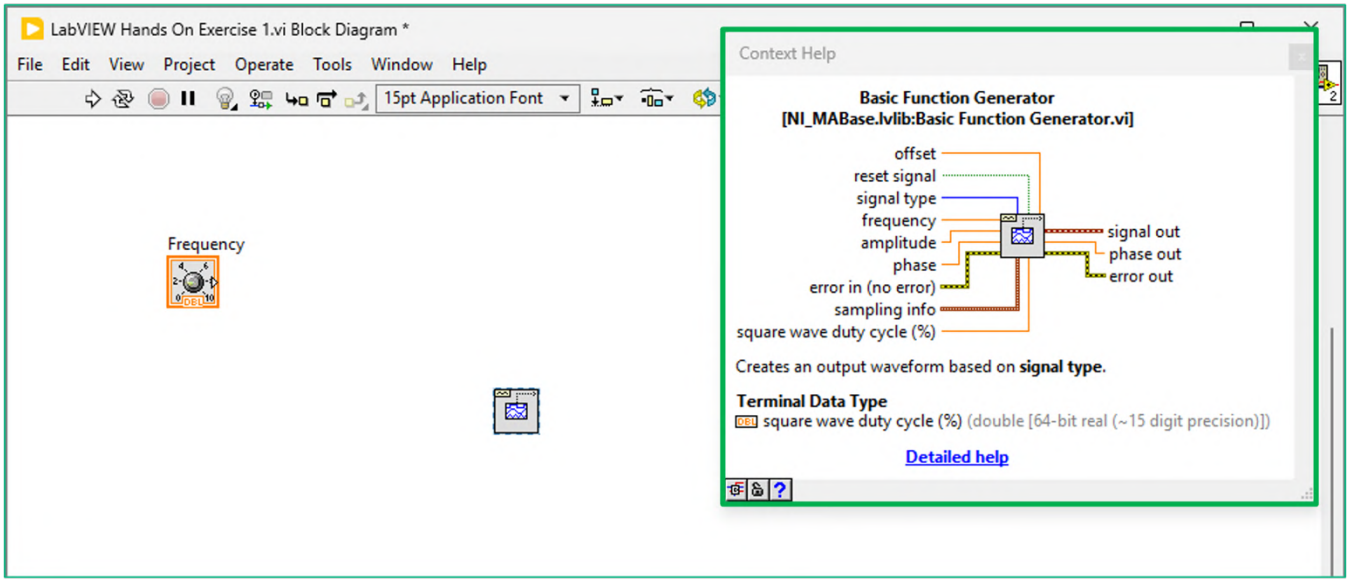


Figure 4. Context Help is a quick way to get a VI's overview, inputs, and outputs.

4. Wire the code to create data flow.

Exercise Note:

To create the dataflow, we need to connect our control, function, and indicator together.

5. Hover your cursor over the **Frequency** control. When the node on the right appears, click it then move your cursor to the top input node on your **Basic Function Generator** function and click it to connect a wire between the control and the function.

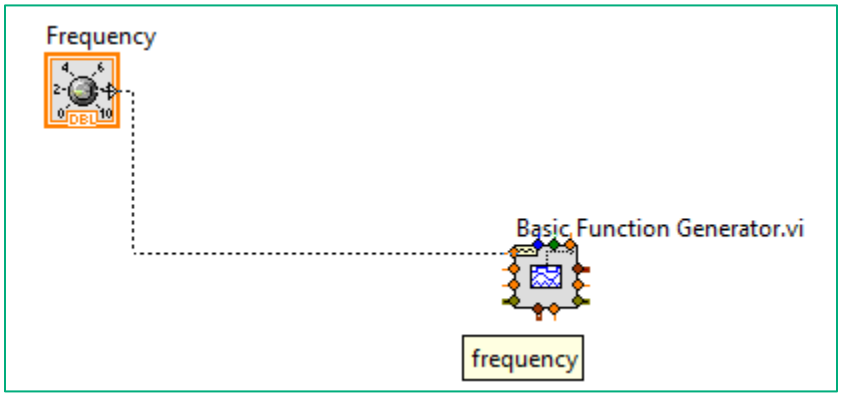


Figure 5. When you hover your cursor on each function, input and output nodes appear. In this example, the knob control will have one output node and the basic function generator has several input and output nodes.

6. Connect a wire from the output of **Basic Function Generator** to the **Time-Domain Signal** waveform graph.

7. Move the **Frequency** control and **Time-Domain Signal** graph to eliminate any unnecessary wire bends.

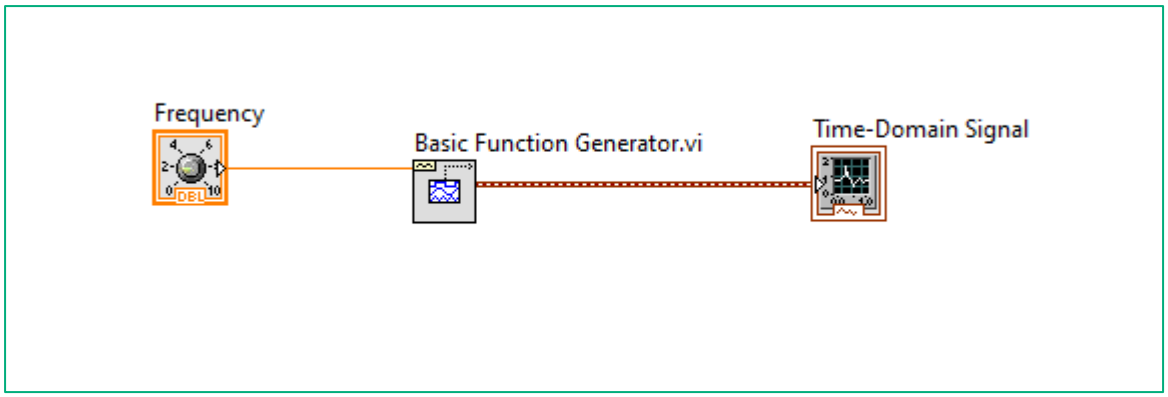


Figure 6. The Basic Function Generator **will receive the** Frequency **input and generate a sine wave that will be displayed on the Time-Domain Signal** graph.

8. Move the **Frequency** knob to 32 (or type out 32 in the digital display), then click the **Run** button. The program will run, and you will see a sine wave with a frequency of 32 Hz on the graph display.

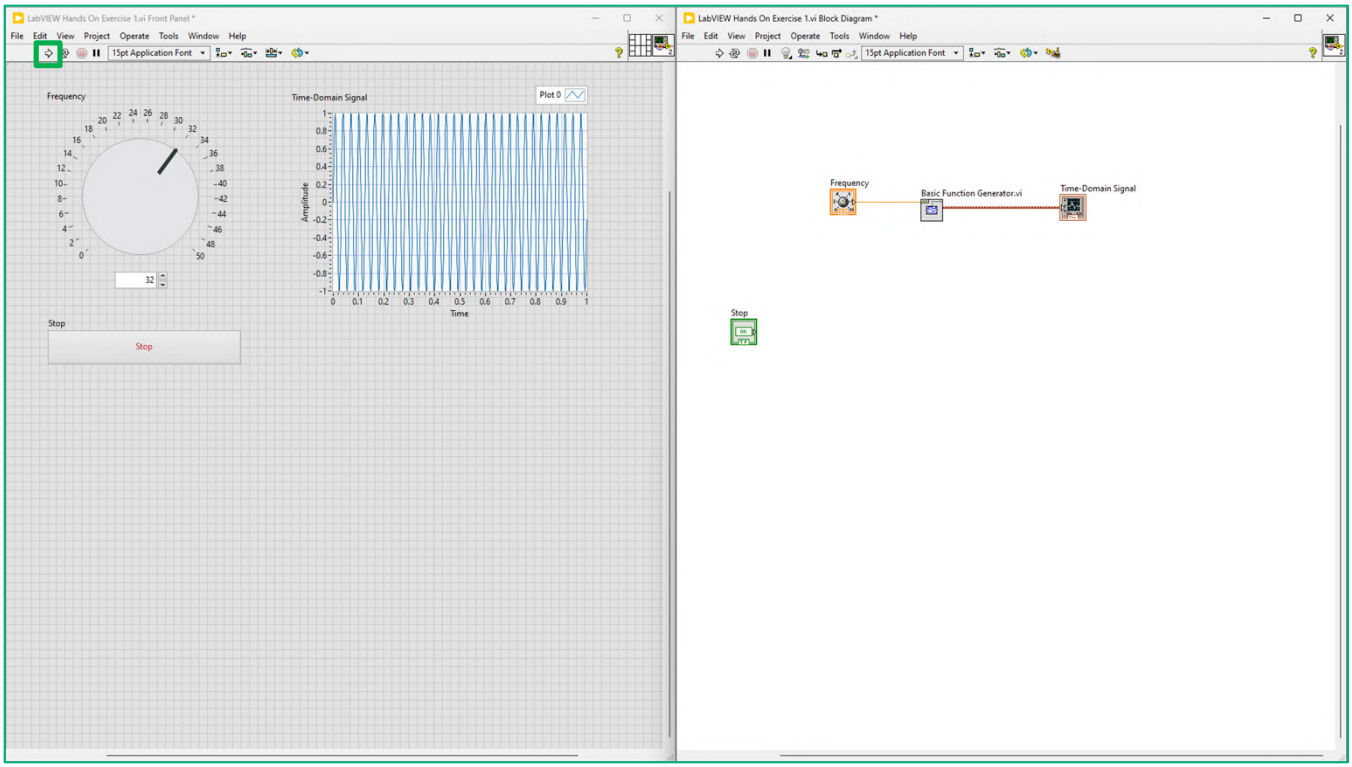


Figure 7. A sine wave with a frequency of 32 Hz will be shown on the graph.

If you change the input to produce another sine wave with a different frequency, you will need to run the application again. This application doesn't run continuously.

Part D: Add the While Loop

Estimated time: 5 min.

In Part D, will make the program run continuously so that you change the frequency instantly see the new sine wave. To do this, we will use a While loop.

1. Right-click on the Block Diagram go to **Programming** >> **Structures** and click on **While Loop**.

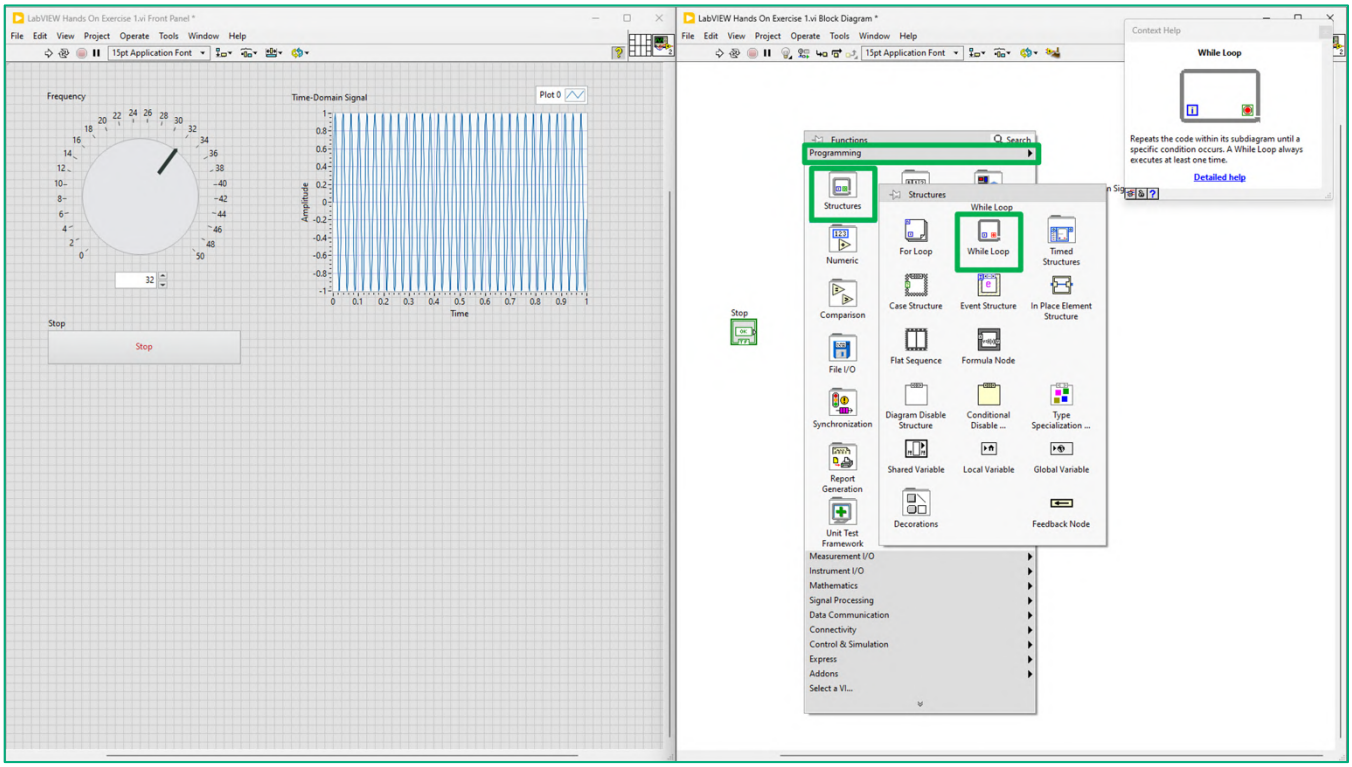


Figure 1. Select While Loop **from the Structures Palette.**

LabVIEW Concept

Structures control the execution of your application. LabVIEW includes for loops and while loops, which you might already be familiar with. Case structures are like “if” statements that will execute a set of code based on a specific condition. There is an event structure, case structure, and more. You can use Context Help to learn more.

2. Click and hold to begin placing it on the Block Diagram. Drag the loop to encapsulate all the existing code. If you made a mistake or want to try again you can Undo by typing **Ctrl+Z**.

3. Now, review the **Run** button, found on the left edge of the menu bar.

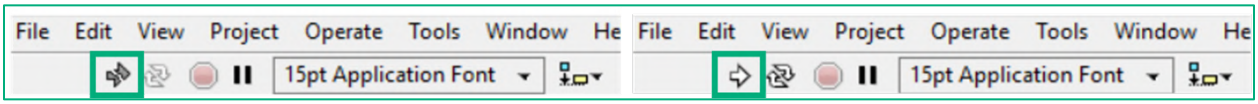


Figure 2. A broken run arrow (left) indicates that there is an error compiling the application.

LabVIEW Concept:

You must press the **Run** button to begin any LabVIEW application. A broken run arrow tells you that there are some unresolved errors in the code. Since LabVIEW is continually compiling code throughout development, you can press the broken **Run** button at any time and a list of current errors will appear.

4. On the bottom right corner of the loop, you'll find a **Loop Condition** terminal, which determines when the loop stops running. We must add a condition to correct the broken **Run** button. In this exercise, we want to give the user control.

5. Wire the **Stop Button** to the **Loop Condition** terminal.

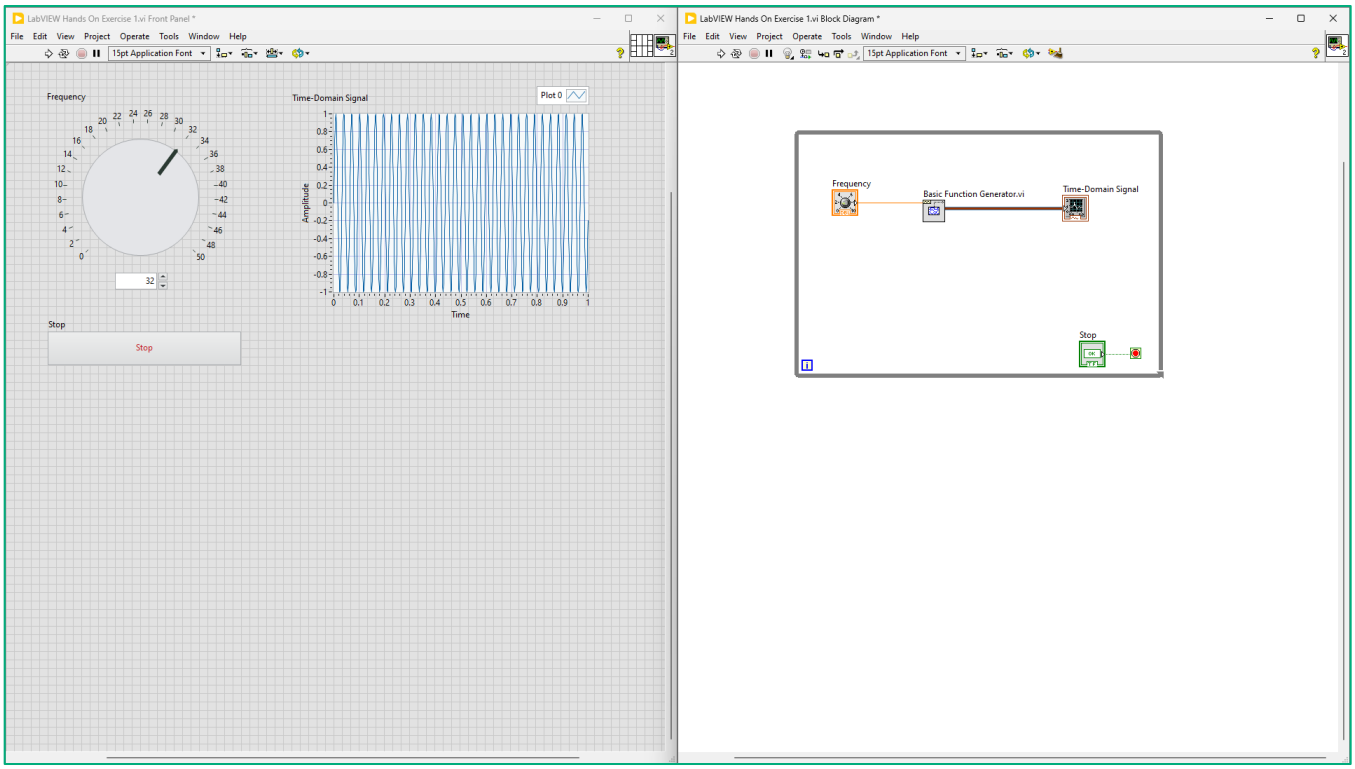


Figure 3. Wire the Stop **button to the** Loop Condition **terminal.**

6. Click the **Run** button on the Front Panel toolbar.

7. Move the **Knob** control on the Front Panel and you can see the corresponding sine wave displayed on the Time-Domain Signal waveform graph

8. After you are done, click the **Stop** button to end the program.

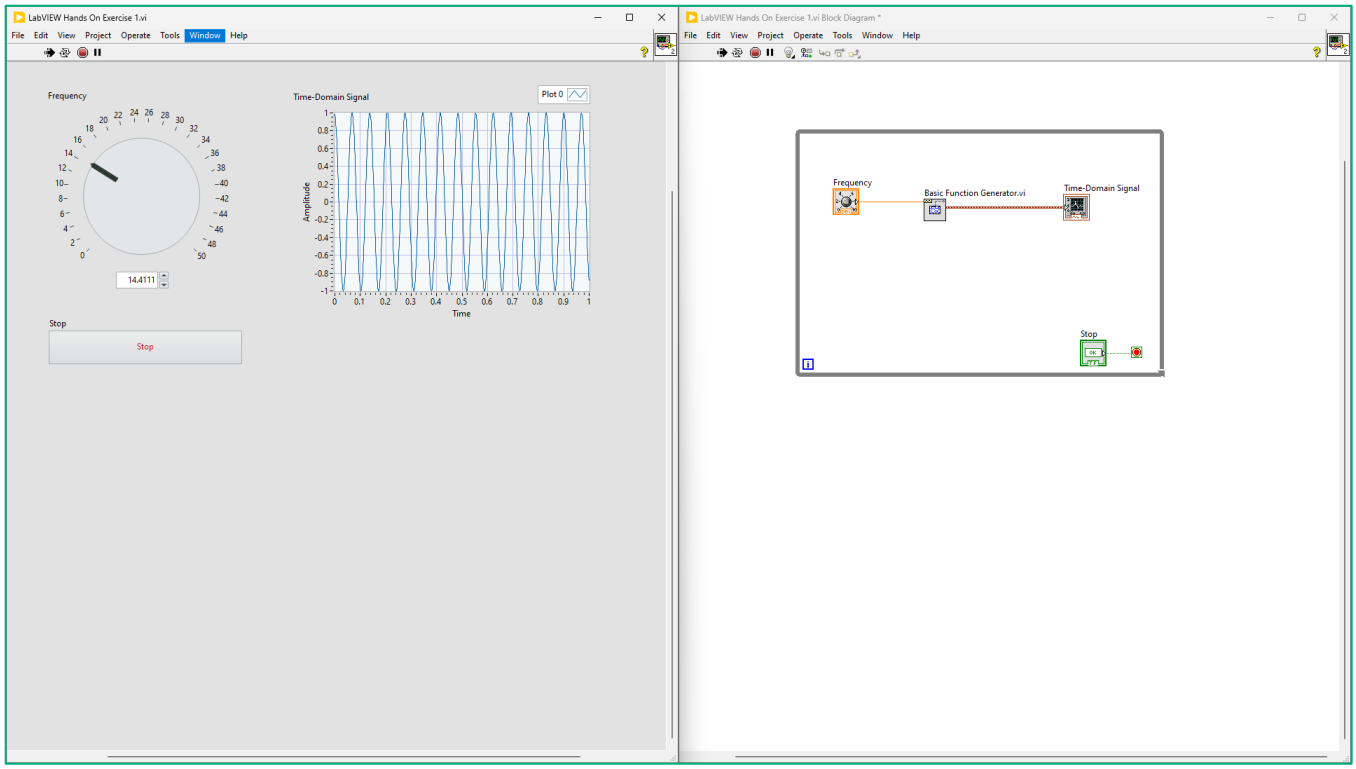


Figure 4. With the While Loop**, your program will now run continuously so you can change the frequency and immediately see an update on the graph.**

9. While changing the Frequency, the graph updated immediately. The application is running as fast as the CPU will allow. It's best practice to introduce a delay between iterations, so you don't overwhelm your processor. Right-click on the Block Diagram **Programming >> Timing** and click on **Wait (ms)**. Place this inside the While Loop.

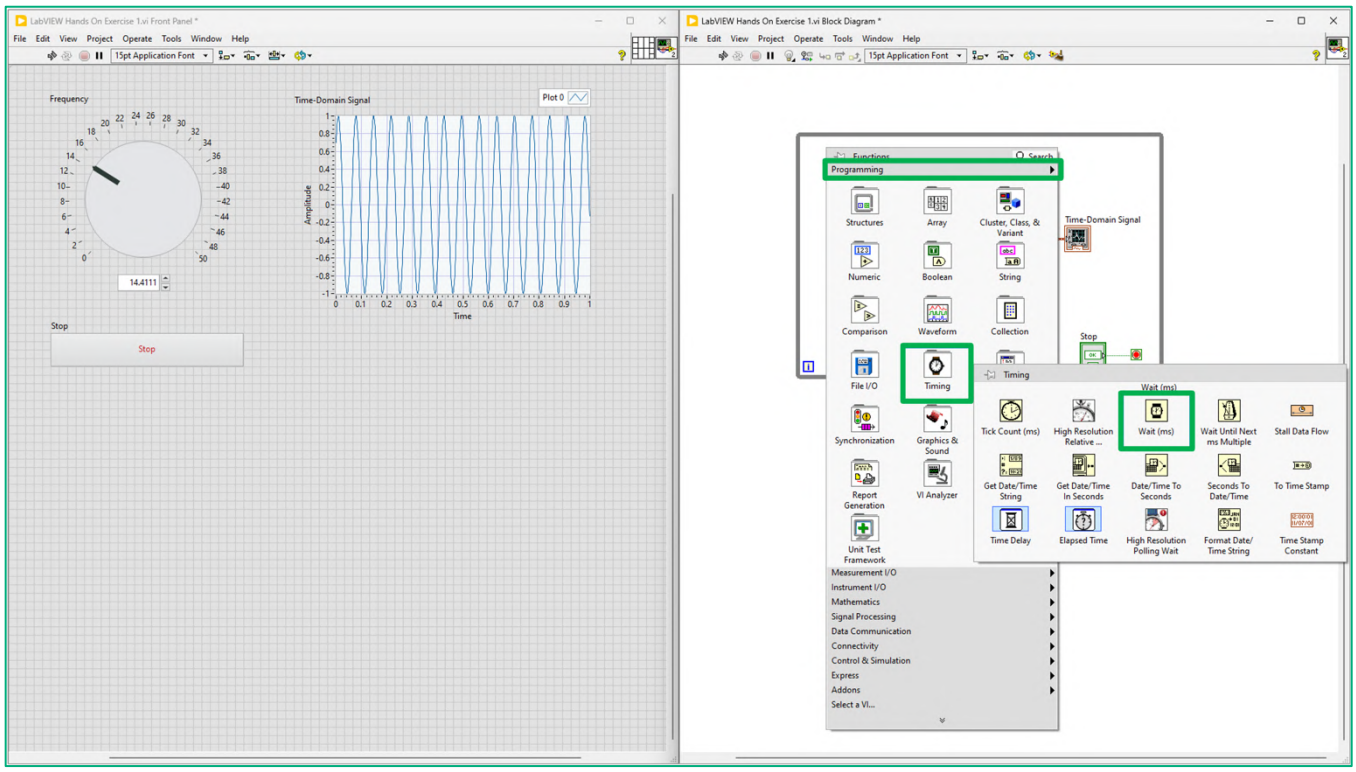


Figure 5. Add a Wait **function inside the While Loop.**

10. Hover over the left side of the function with your cursor. Notice the wiring tool appear. Right-click on **Wait** function and go to **Create** click on **Constant**. Type in **100**. The program will now run every 100 ms.

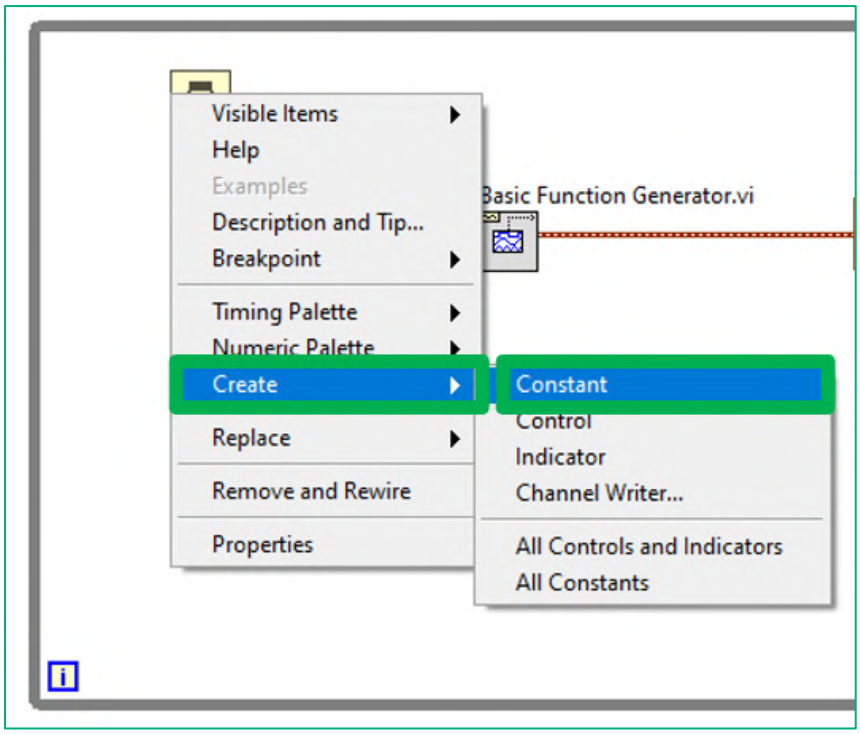


Figure 6. Create a Constant **for the Wait function. Set it to** 100 ms.

11. Right-click on the **Wait** function, again. Go to **Visible Items >> Label**.

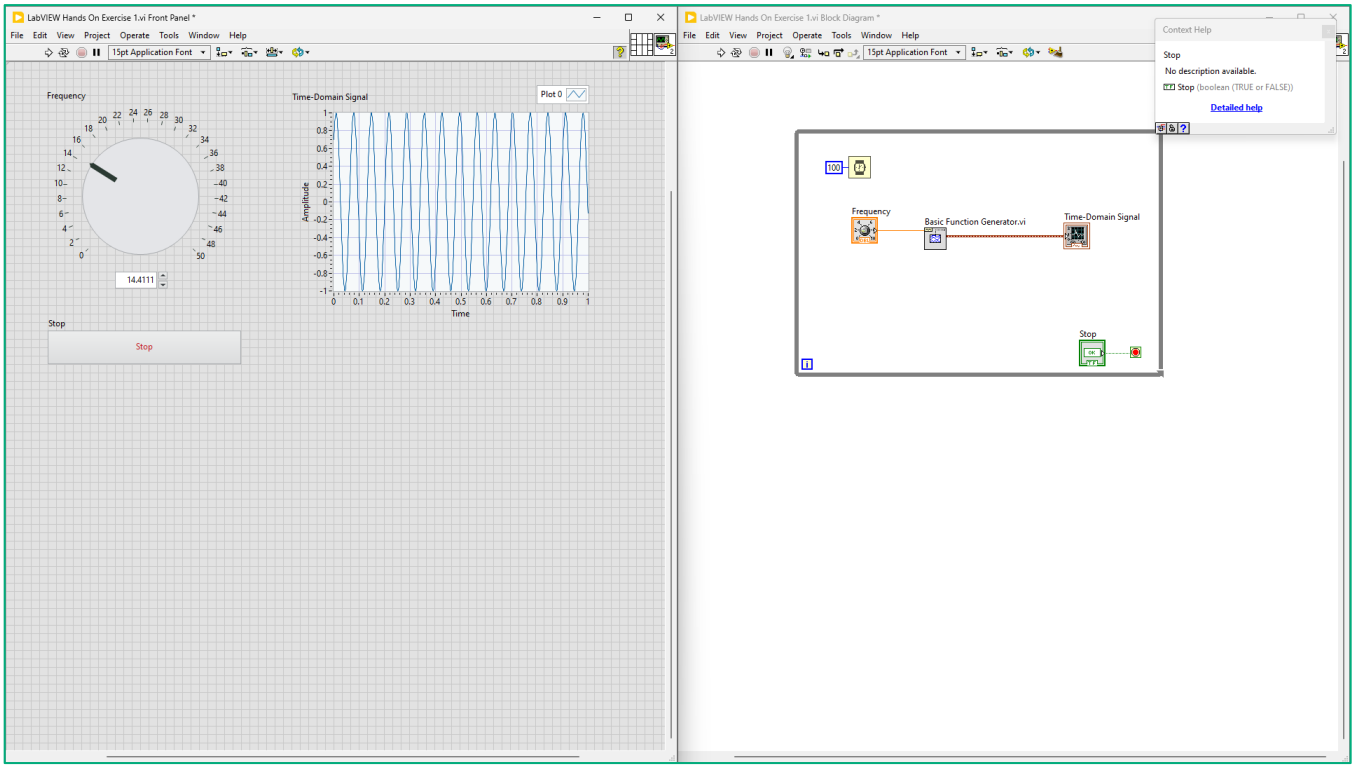


Figure 7. The completed application should look like this.

12. Click the **Run** button on Front Panel toolbar. Notice the difference in the execution speed. Press the **Stop** button when you are finished.

Exercise Key Concepts:

In this exercise, we learned about the LabVIEW environment, the Front Panel and the Block Diagram, and the basics of dataflow programming while creating a simple interactive application. Building an interactive user interface in other programming languages can take time, but with LabVIEW, you were able to do it in moments by dragging-and-dropping – and without any advanced programming skills.

<End of Exercise>

Bonus: Tips and Tricks with LabVIEW!

Estimated time: 5 minutes.

LabVIEW provides several tools that can help you develop your applications. The next few steps will show how to use some of the most important programming assistance tools.

Block Diagram Cleanup

As you program, and especially as you learn how to program in LabVIEW, you are not always thinking about layout and readability. This can result in a poorly organized Block Diagram.

LabVIEW's Block Diagram Cleanup is a built-in tool that organizes your code, making it easier for you and others to understand how your program functions.

1. Press the **Block Diagram Cleanup** icon found on the menu bar.

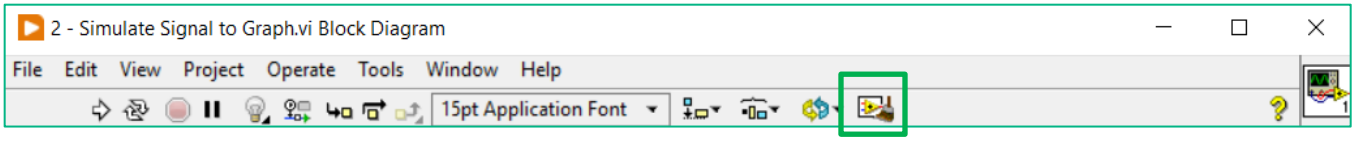


Figure 1. Block Diagram Cleanup will automatically organize your code.

Your Block Diagram should now be organized, with cleaner wires and an even distribution of code elements.

Highlight Execution

1. Press the **Highlight Execution** button on the menu bar. Notice that the light bulb icon now appears to be on.

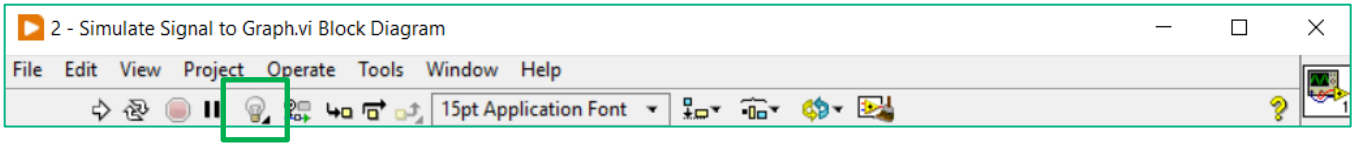


Figure 2. Highlight Execution slows your code to let you visualize dataflow.

2. Run your application with Highlight Execution turned on. Press the **Run** arrow and watch as your code executes step-by-step. While not always necessary for simple applications, the Highlight Execution tool is a powerful tool for troubleshooting complex programs and determining if your code performs as expected.

Context Help

1. Press the **Context Help** button in the upper right portion of the Block Diagram.

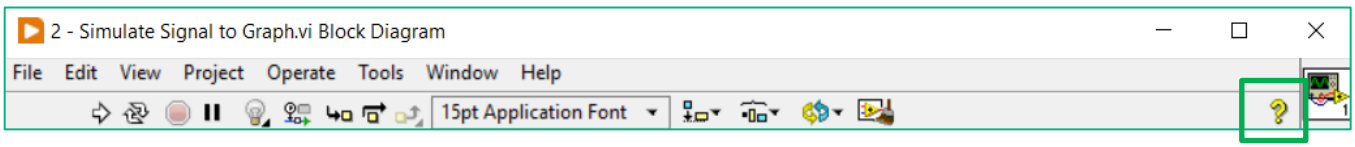


Figure 3. Context Help displays a context-based help dialog.

2. With Context Help active, hover your cursor over different objects on the Block Diagram and Front Panel of Simulate Signal to Graph.vi. As you do so, the Context Help Window provides details including descriptions and wiring diagrams.

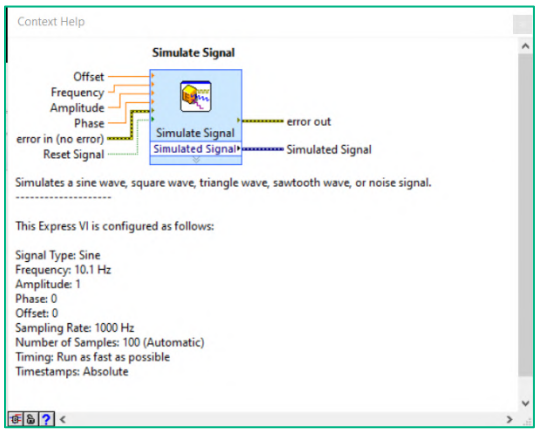


Figure 4. Context Help is a quick way to get a VI's overview, inputs, and outputs.

3. Right-click the Block Diagram and navigate around the palettes. Notice that the Context Help window provides details on the objects while they are in the palettes. Also notice that for some objects, the Context Help window provides a link for Detailed Help. This link will open the *LabVIEW Help* and give you more information.

Quick Drop

1. **<Ctrl+Space>** brings up the Quick Drop dialog. Rather than searching for a function, you can easily locate it with quick drop! Try by typing "While Loop" and it will appear in the list of possible objects.

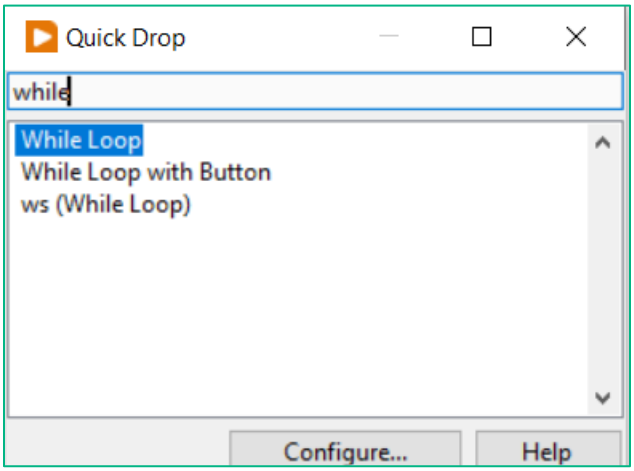


Figure 5. Quick Drop is a *quick* way to find functions and add them to your Block Diagram.

<End of Exercise>

Exercise 2: Take a Voltage Measurement

Estimated time: 45 minutes.

Goals:

- Learn how to acquire data from a CompactDAQ device with LabVIEW.
- Build an application that acquires data, performs analysis, and saves raw data to a file.

Part A: Acquire Data using the DAQ Assistant

Estimated time: 20 minutes.

In Part A, you will use LabVIEW quickly set up a program to acquire voltage data from the simulated device.

1. If you would like to save your progress, go to **File >> Save As...** The setting **Substitute copy for original** will be selected. This will save a copy of your work and allow you to save a new file to work in. Press **Continue…** Save the file as **"LabVIEW Hands On Exercise 2"**

2. We will be replacing the Basic Function Generator with data from a simulated device. Right-click on the Block Diagram. Go to **Measurement >> NI DAQmx** and click on **DAQ Assist.** Place this inside the While Loop.

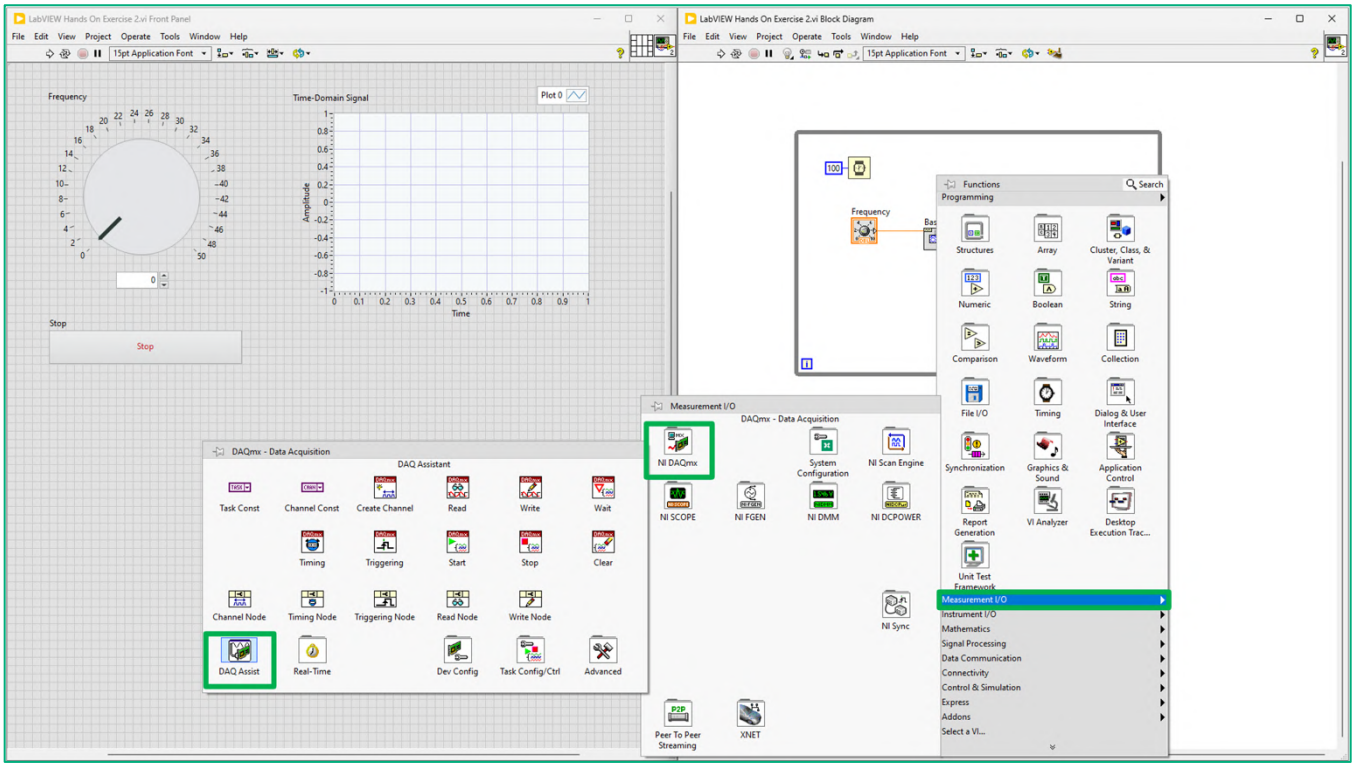


Figure 1. The DAQ Assistant is a configuration-based way to acquire data from physical or simulated NI DAQ hardware.

3. Once the **DAQ Assist** is placed, the **Create New**… window appears. This might take a few seconds to initialize.

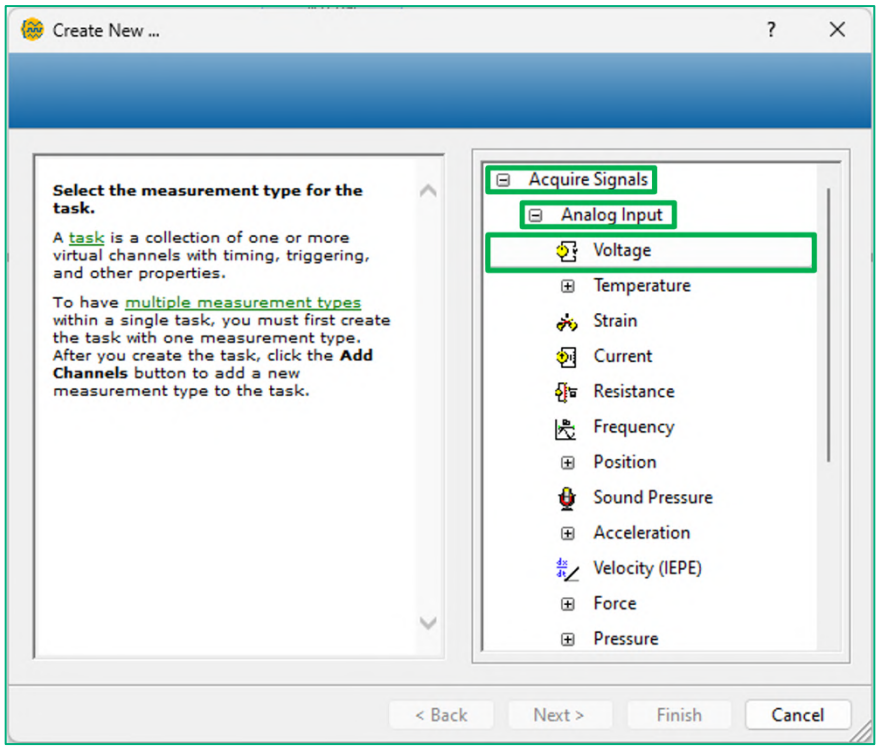


Figure 2. NI-DAQmx tasks are a collection of channels with homogenous timing and triggering properties.

4. To configure a voltage measurement, click **Acquire Signals » Analog Input » Voltage**

5. Click the **+** sign next to the module named **Voltage_in (NI 9215)** highlight channel **ai0**, and click **Finish**. This adds a physical channel to your measurement task.

6. In the **Timing Settings**, change the **Samples to Read** to **1k** and change the **Rate (Hz)** to **1k**.

7. Click the **Run** button. You will see the voltage readings in test panel window.

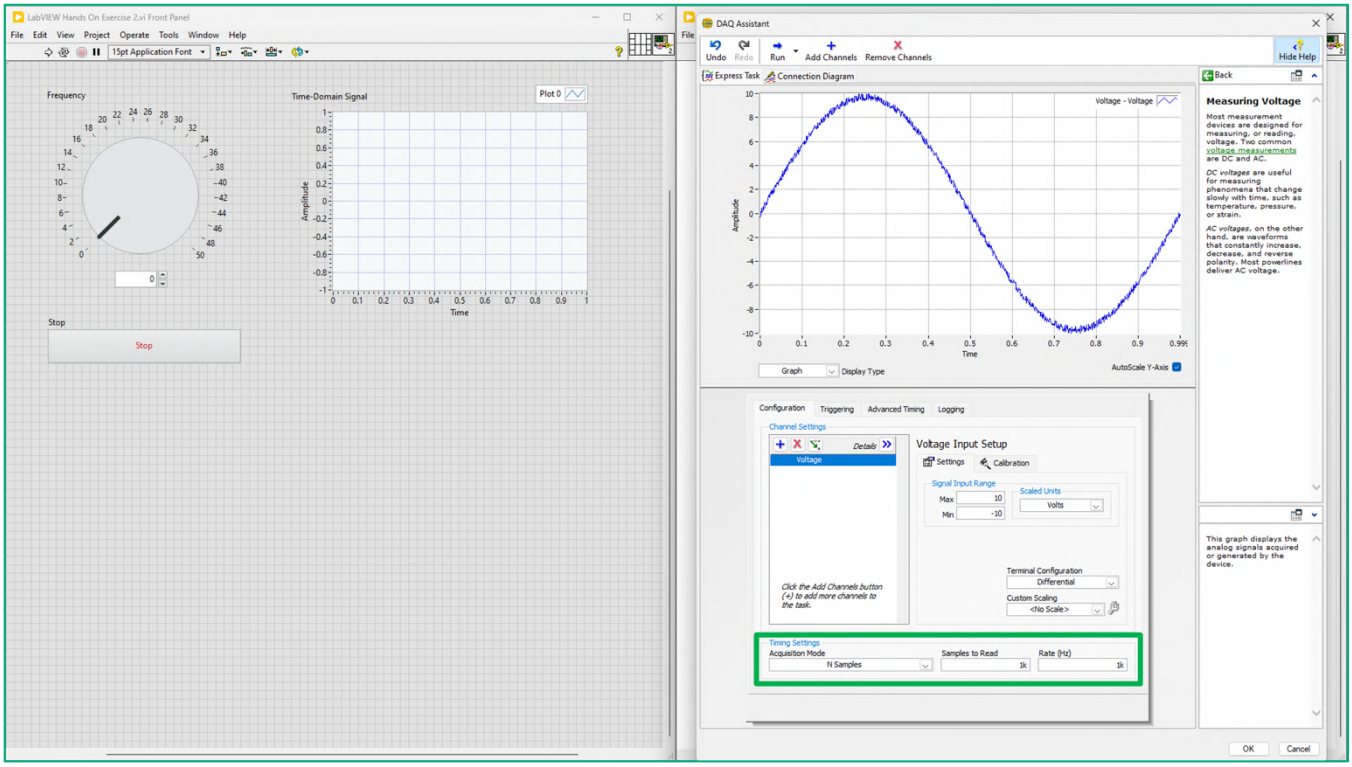


Figure 3. You can configure and test your measurement from within the DAQ Assistant window.

8. Click **OK** to close the DAQ Assistant wizard and return to the Block Diagram. LabVIEW automatically creates the code for this measurement task.

9. We will now replace the Basic Function Generator with the DAQ Assistant. Click on **Basic Function Generator** and hit **Backspace** or **Delete** on your keyboard.

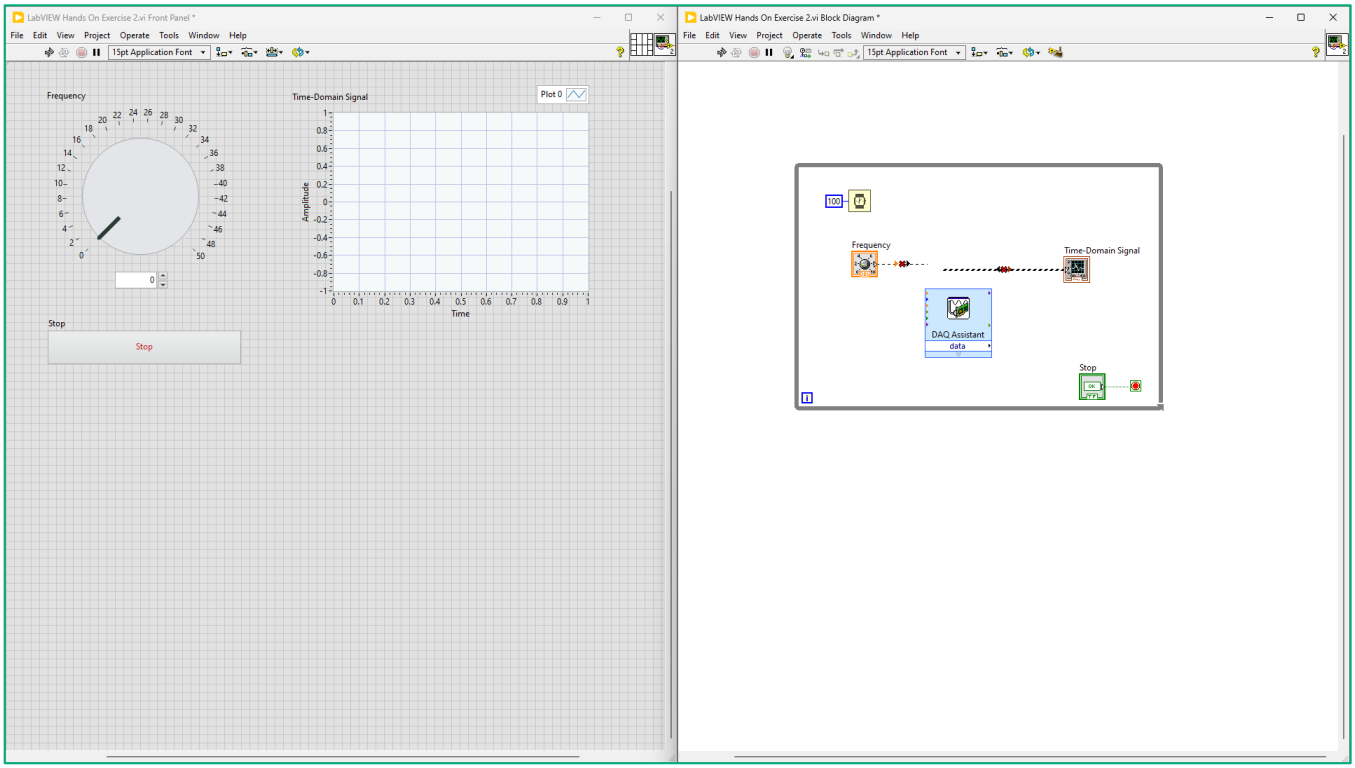


Figure 4. Delete the Basic Function Generator**. The wires that were previously connecting the functions become broken.**

10. Move the **DAQ Assistant** up in the Block Diagram, towards where the Basic Function Generator was. Type **Ctrl+B** to remove the broken wires.

11. Click the output node of the **Frequency** knob and wire it to the top input node of the **DAQ Assistant.**

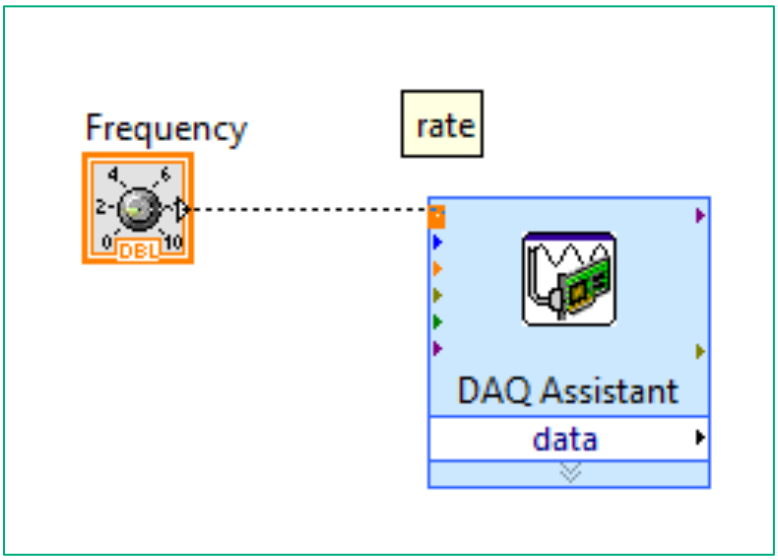


Figure 5. Wire the Frequency **knob to the** DAQ Assistant.

12. Wire the data output terminal on the **DAQ Assistant** to the **Time-Domain Signal** waveform graph.

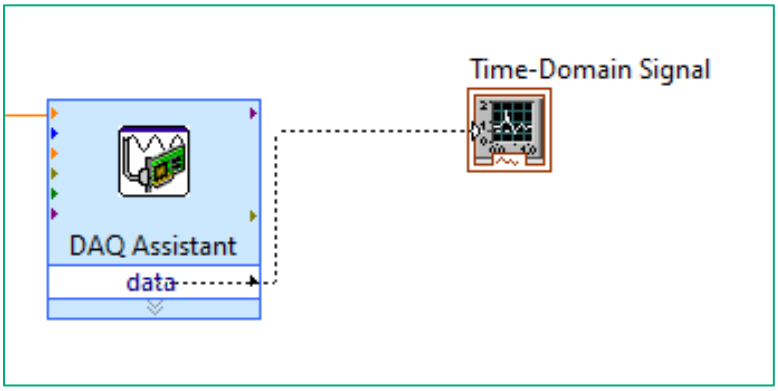


Figure 6. Wire the DAQ Assistant **to the** Time-Domain Signal**.**

13. Move the **Time-Domain Signal** block down to align with the output and eliminate unnecessary wire bends.

14. Right-click the second input terminal on the **DAQ Assistant** titled Number of Samples. Go to **Create** and click **Constant**. Type in **1000**.

15. Go to the Front Panel and double-click on the knob label **“Frequency”** and change it to **“Acquisition Rate”**.

16. Change the scale for the **Acquisition Rate**. Double-click on the first number, **“0”,** and type in **“20000”**. Double-click on the last number, **“50”**, and type in **“30000”**.

17. Move the **Acquisition Rate** to about **27500**.

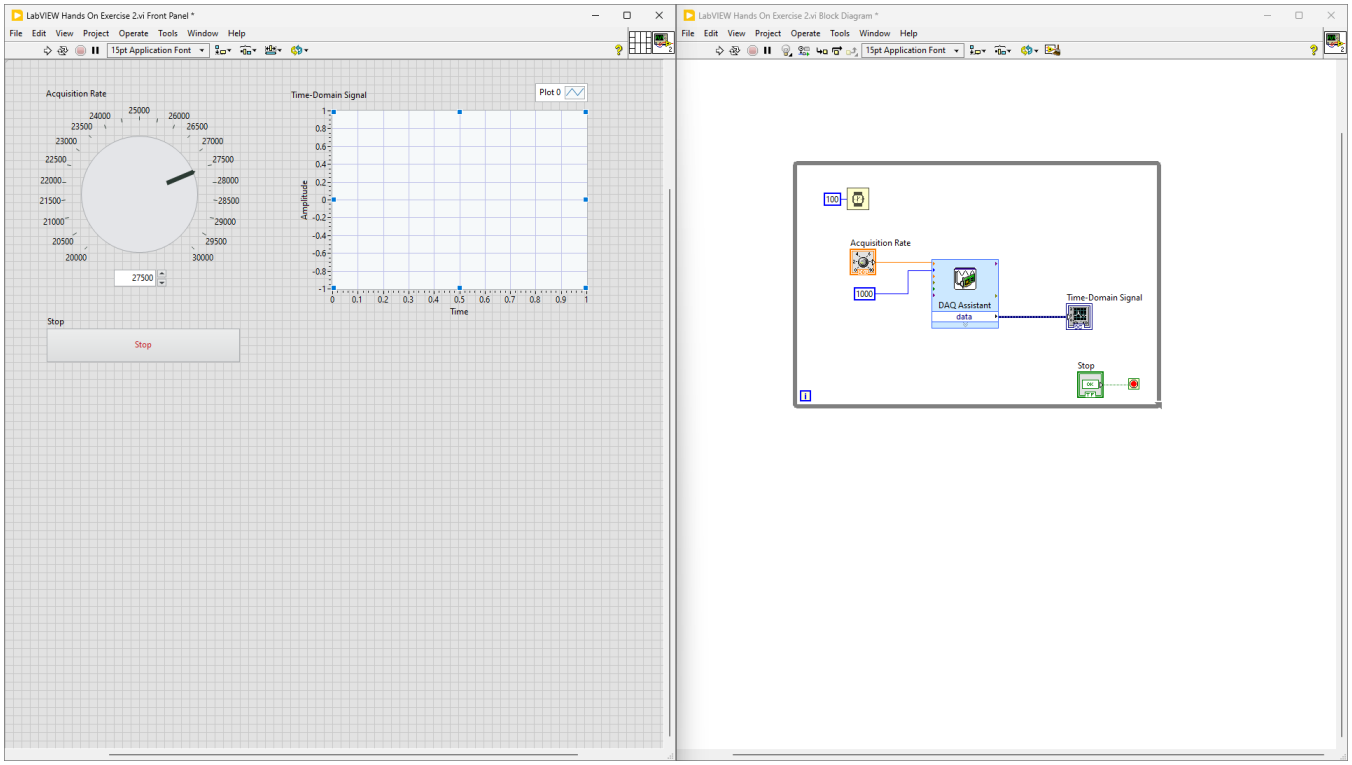


Figure 7. The application can now acquire data from a NI DAQ device and display it on the graph.

18. **Run** the VI. Change the **Acquisition Rate** as desired. Press the **Stop** button when you are finished.

Exercise Note:

Express VIs make creating basic tasks very easy. Their configuration dialogs allow you to set parameters and customize inputs and outputs based on your application requirements. They are good starting points, but as you continue to use LabVIEW and increase your skill, you may need more than the Express VI can provide. If you want to optimize your DAQ application’s performance, you should use the standard DAQmx driver VIs. The DAQ Assistant can automatically generate standard DAQmx code; not all Express VIs have this functionality.

Part B: Analyze the Acquired Data

Estimated time: 15 minutes.

In this Part B, we will add an analysis function to our program.

1. On the Front Panel, click on the **Time-Domain Signal** waveform graph. Type **Ctrl-C** to copy the graph, just like you would text in a document. Click out of the graph anywhere on the Front Panel. Then, type **Ctrl-**V to paste the duplicate graph.

2. Move the **Time-Domain Signal 2** graph bellow the **Time-Domain Signal** graph.

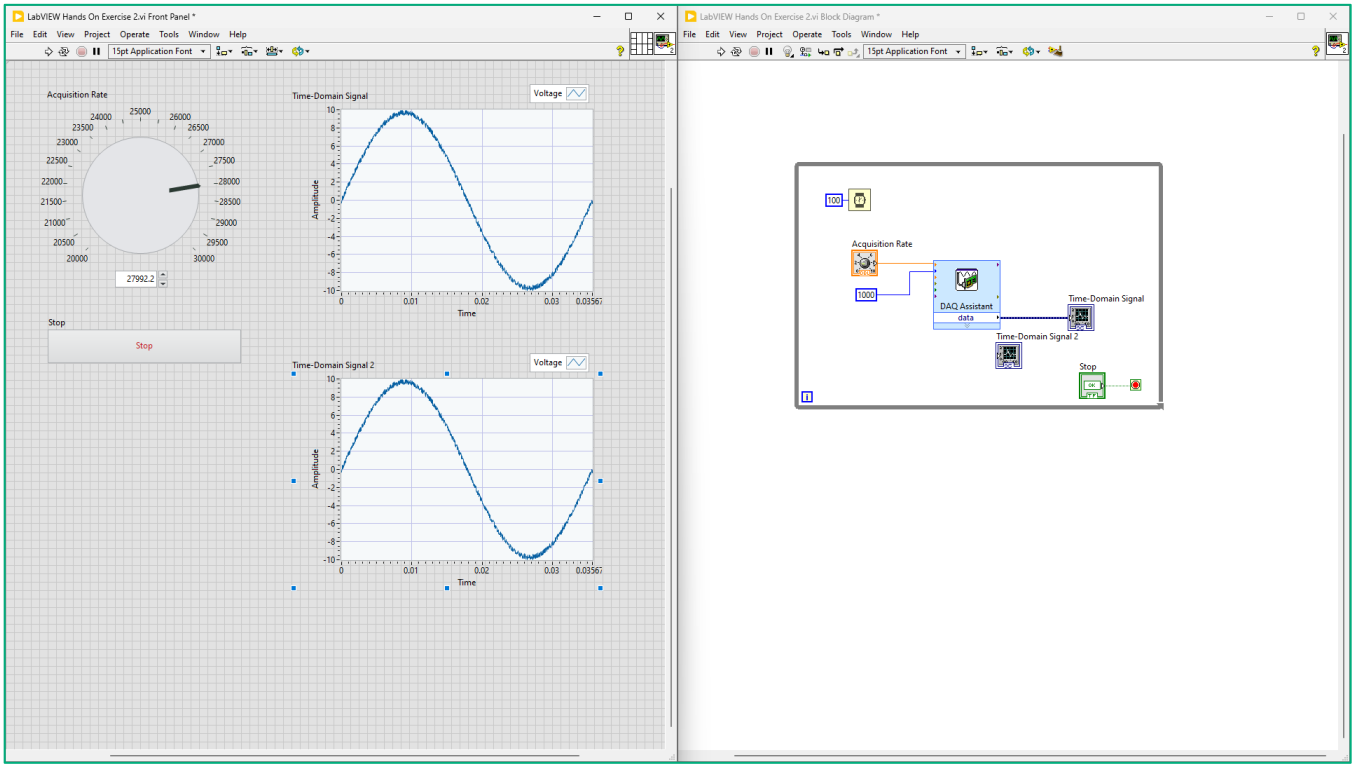


Figure 1. Add a second graph to your Front Panel by using copy and paste. This is a fast way to duplicate an item that you already have in LabVIEW – you can do this with items on the Front Panel and chunks of code on the Block Diagram.

3. Double-click on the label **"Time-Domain Signal 2"** and type **"Frequency Domain Signal"**.

4. We will add the analysis function using **Quick Drop**. Type **Ctrl+Space** to open Quick Drop.

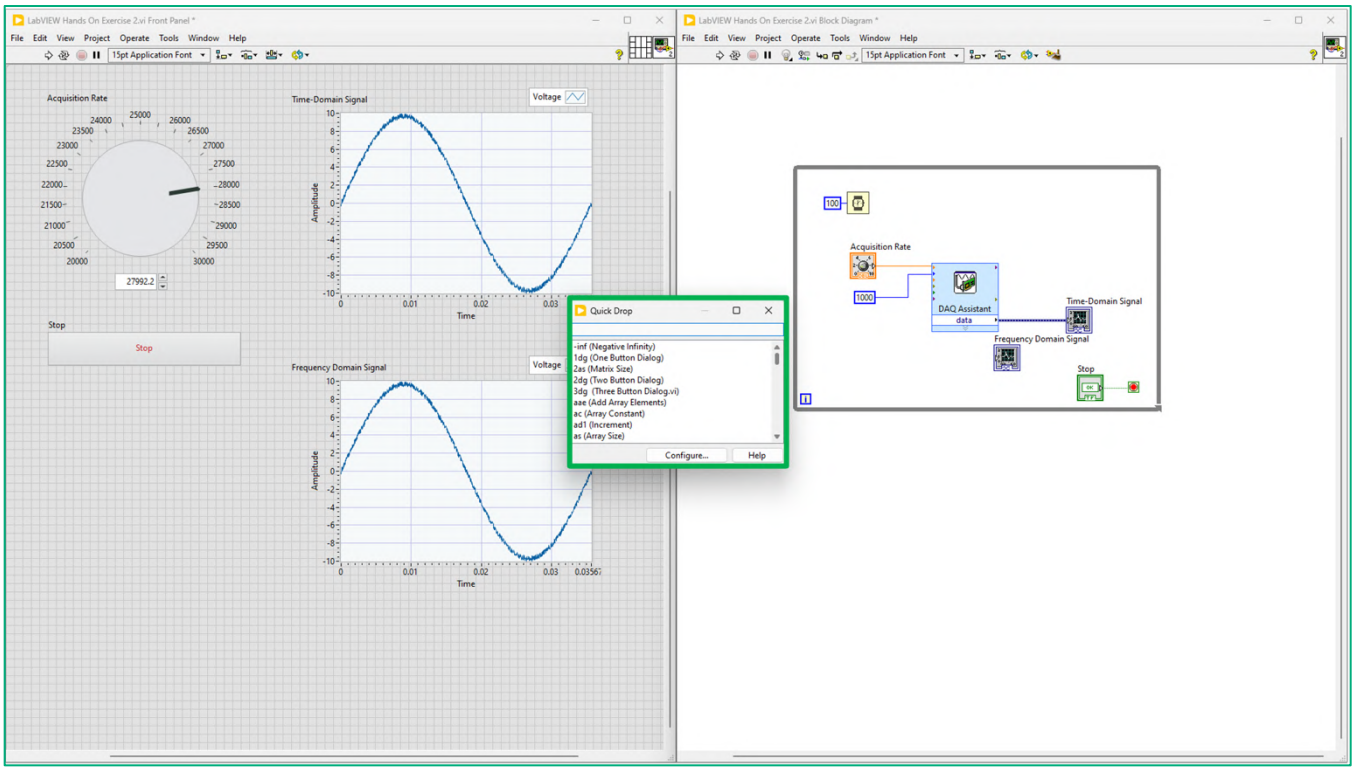


Figure 2. Type Ctrl+Space to open Quick Drop, it is a *quick* way to find functions and add them to your Block Diagram.

5. In **Quick Drop**, type **"fft power"**. Then **Enter** on your keyboard. The **FFT Power Spectrum and PSD** function are ready to be dropped on the Block Diagram. Click anywhere on your Block Diagram to place it.

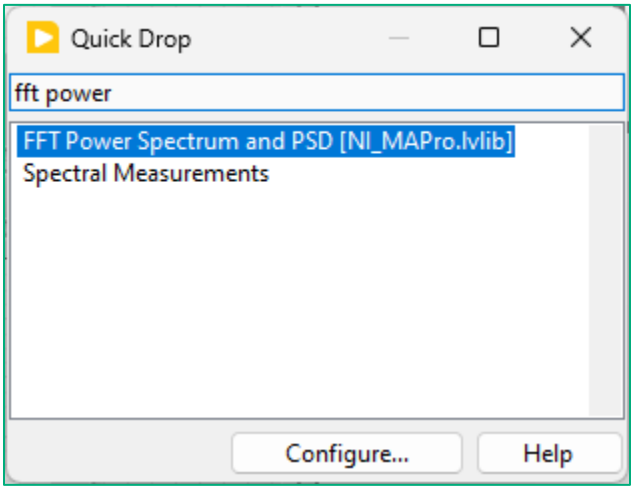


Figure 3. Quick Drop automatically completes queries with the first VI that begins with the same letters and shows all other possible matches in the list.

6. Right-click on the **FFT Power Spectrum and PSD** and go to **Visible Items** and select **Label**.

7. It might be helpful at this point to organize the Block Diagram
.

    a. Click on the **While Loop** and use the resizing handles at the bottom to make it taller and/or wider.

    b. Click inside the **While Loop** just beneath the **Wait (ms)**. Drag your mouse to create a box that encloses the Acquisition Rate, Number of Samples, DAQ Assistant, and Time-Domain Signal. Release. Move the objects lower and slightly to the left.

    c. Move the **Frequency Domain Signal** below the Time-Domain Signal.

    d. Create another box that encloses the **Stop** button and **Loop Condition**. Move both closer towards the bottom right of the While Loop.

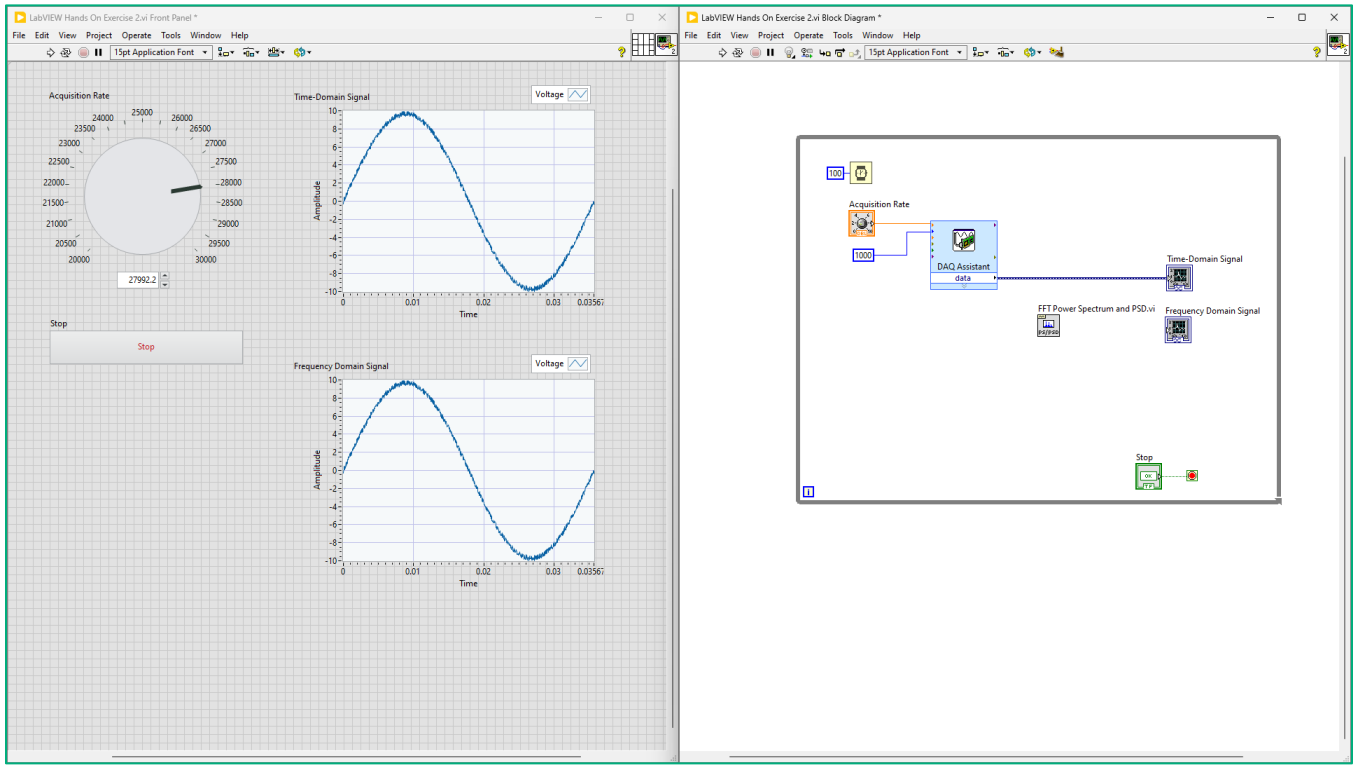


Figure 4. It is helpful to organize the Block Diagram so you can easily view, understand, and modify your application.

8. Click on the top input terminal on the **FFT Power Spectrum and PSD** function labeled **Time Signal** to create a wire to the data output of the **DAQ Assistant**.

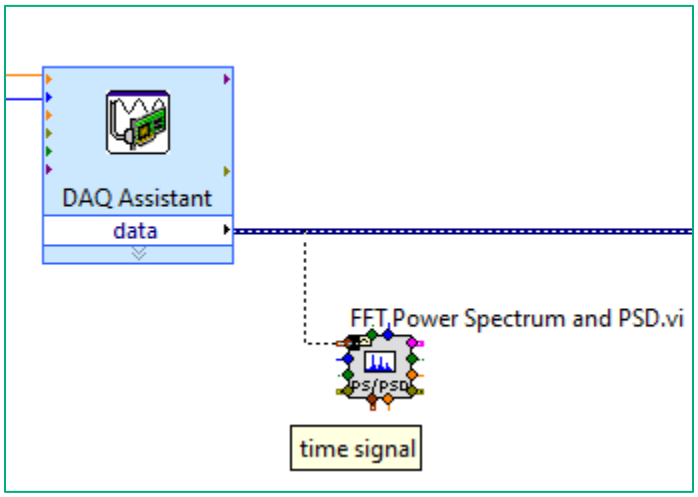


Figure 5. Wire the data output from the DAQ Assist **to the** FFT Power Spectrum and PSD**. You can have an output go to multiple functions – here we have the data going to the graph and the FFT function.**

9. Wire the top output on the **FFT Power Spectrum and PSD** labeled **Power Spectrum / PSD** to the input of the **Frequency Domain Signal** block.

10. Go to the **Frequency Domain Signal** graph on the Front Panel. Right-click near the **X-axis**. Click on **AutoScale X** to turn off autoscaling. Double-click on the last number on the X-axis and type **“50”**.

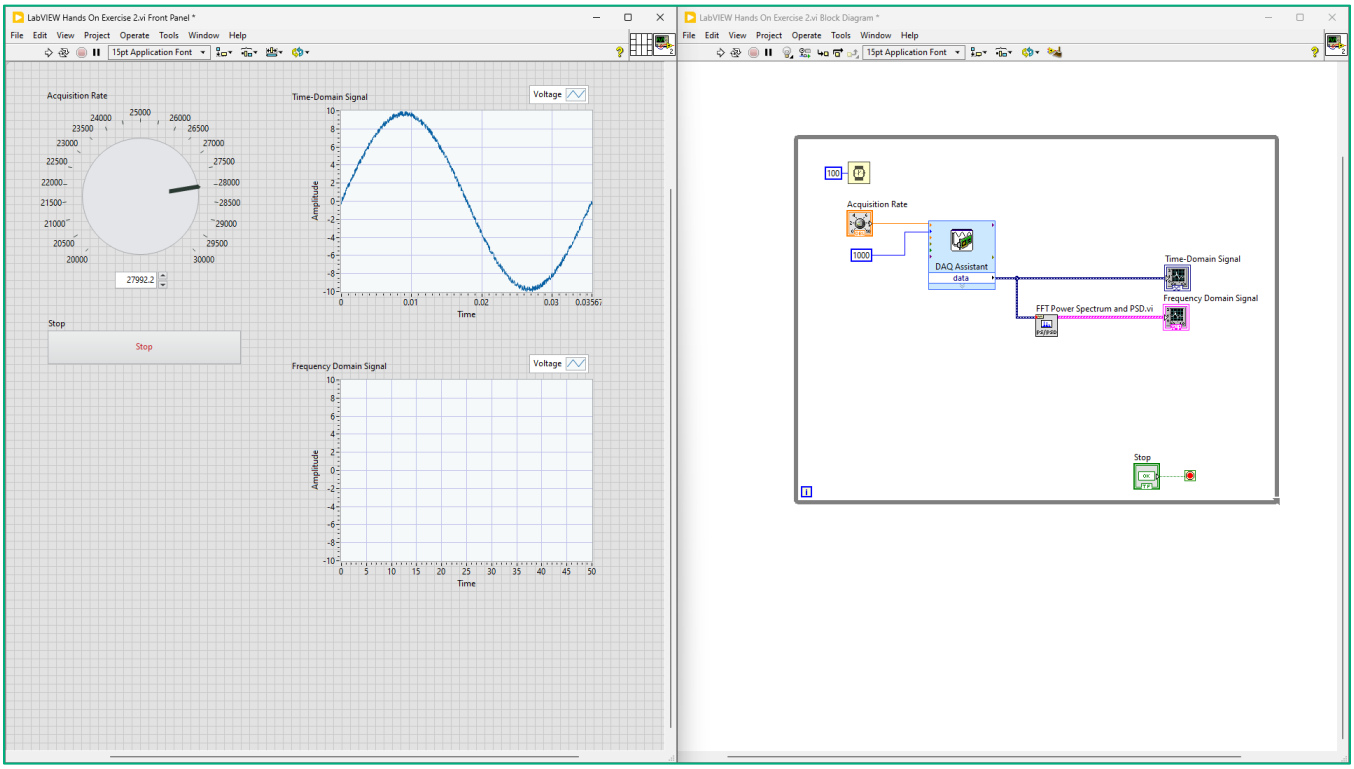


Figure 6. Modify the X-axis on the second graph by using the Properties window or by simply typing in your desired limit, after you turn off AutoScaling.

11. Click **Run** to start the VI. After some time, press the **Stop** button on the Front Panel.

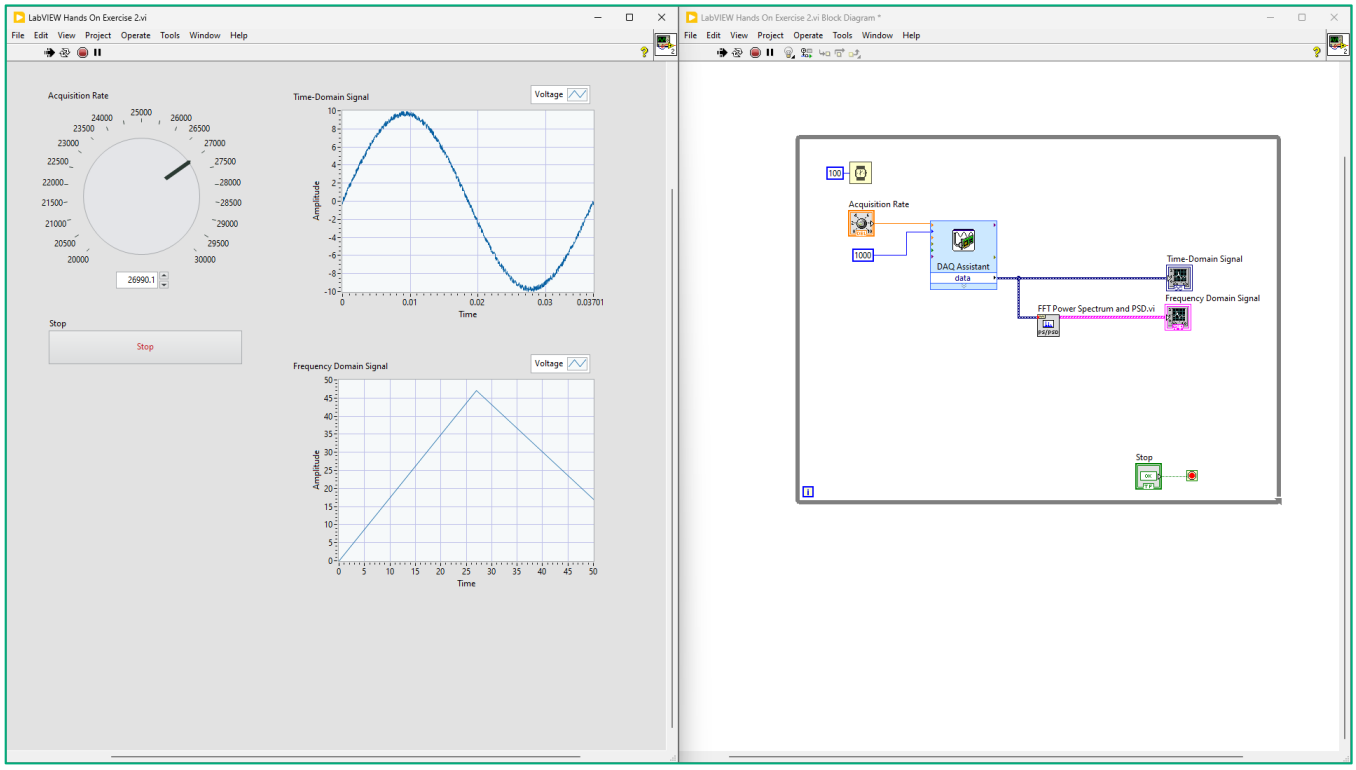


Figure 7. The completed application that acquires a voltage input and computes an FFT on the signal.

Part C: Save the Data to a File

Estimated time: 10 minutes.

In Part C, you will save the acquired data to file so you can share it with colleagues, create a report, run additional analysis, or all the above.

1. Right-click the Block Diagram and go to **Programming » File I/O** and select **Write to Measurement File**. Place the Express VI inside the While Loop on the block diagram.

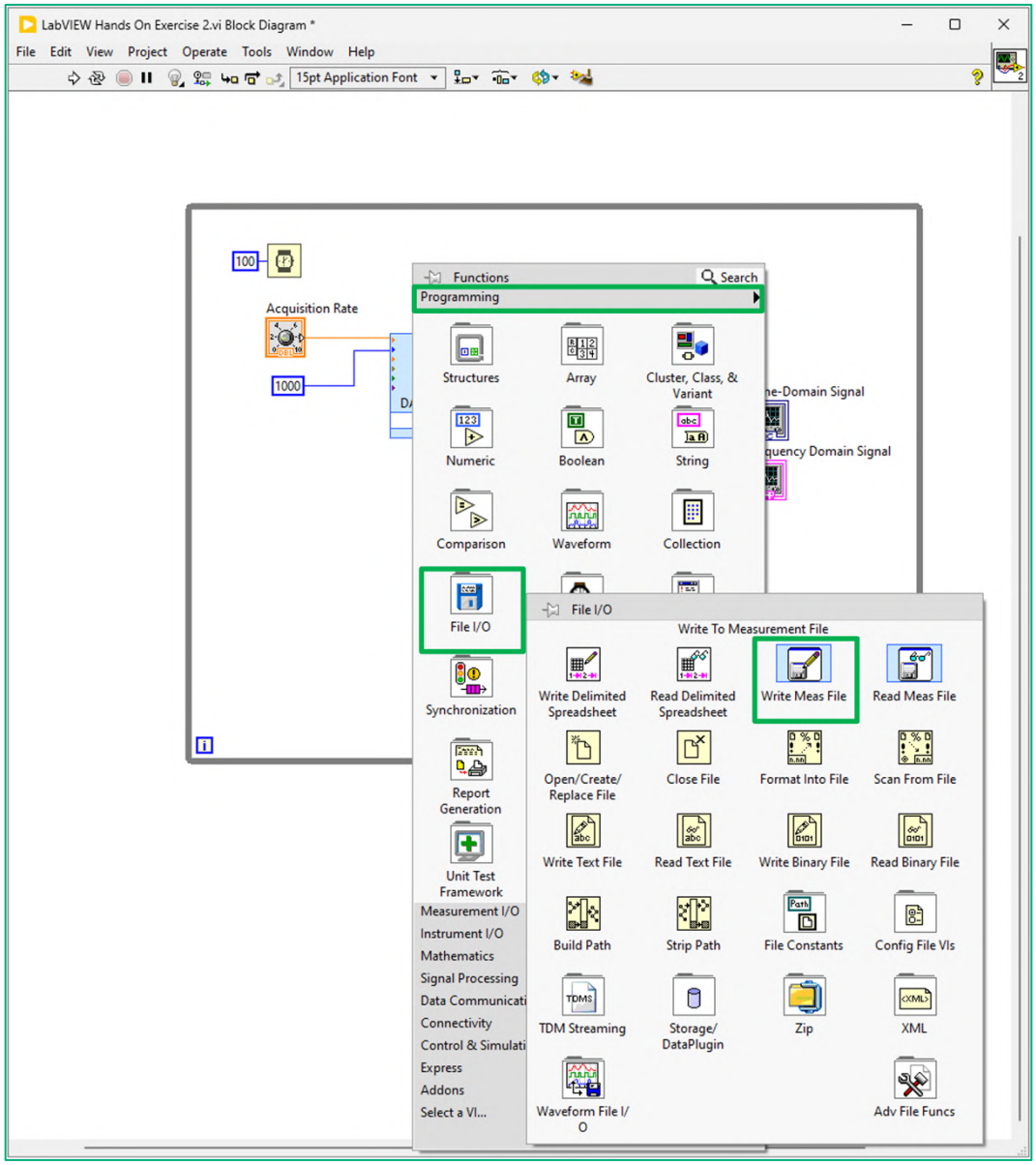


Figure 1. LabVIEW includes dozens of options for reading and writing files. You can also add this using Quick Drop.

2. A configuration window will appear. Configure the window as shown below. When finished, click **OK**.

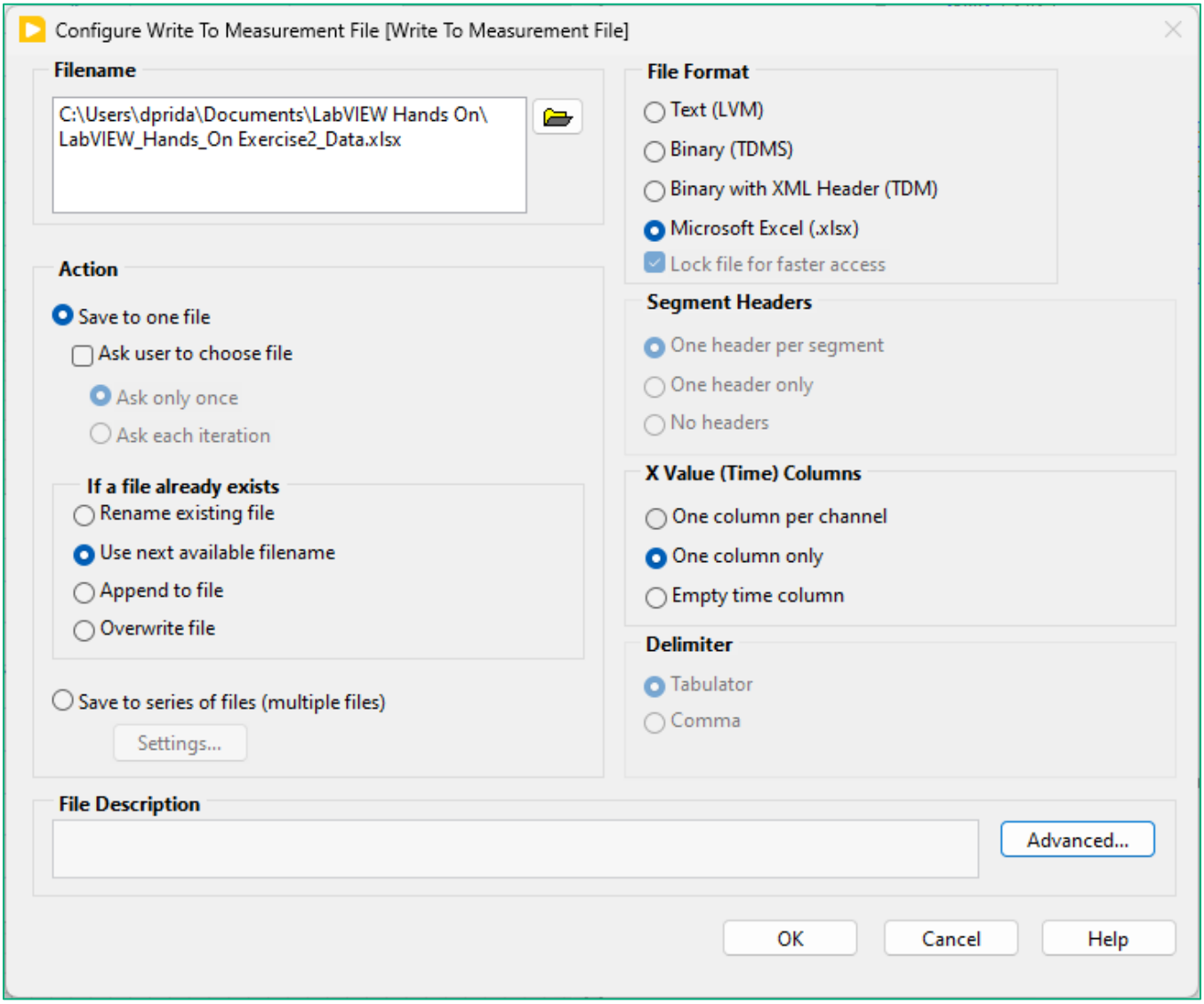


Figure 2. This configuration will save a file that can be opened by Microsoft Excel.

3. You may need to create more space on your Block Diagram by enlarging the **While Loop**. Use the resizing handles just as before.

4. Wire the output of the **DAQ Assistant** to the input of the **Write to Measurement File Express VI**.

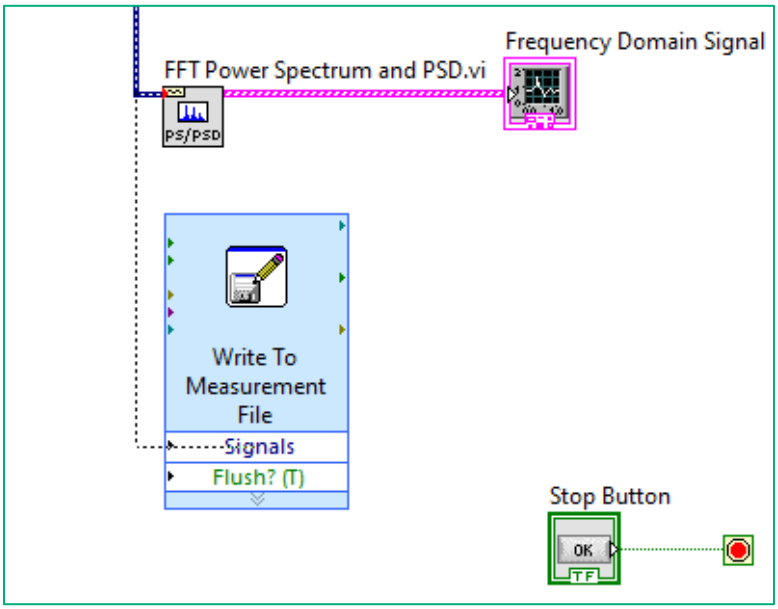
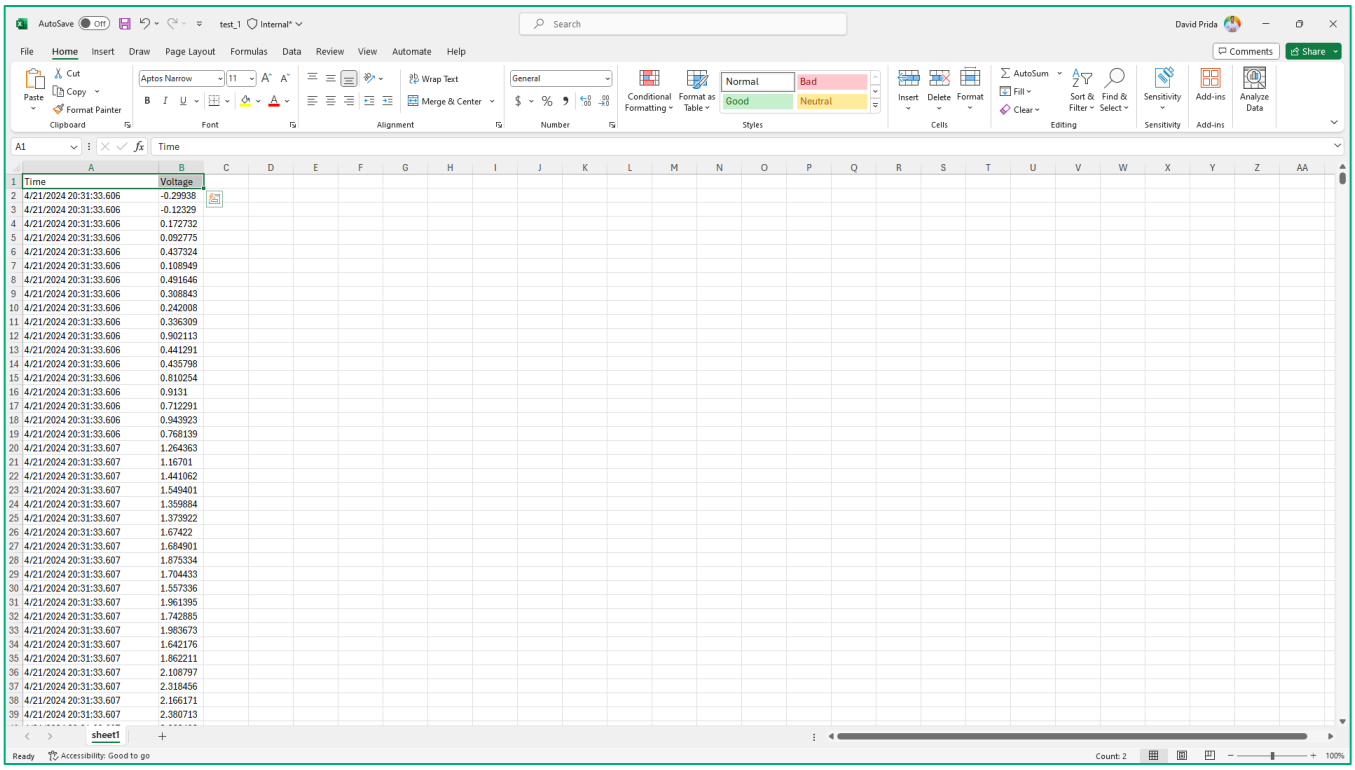


Figure 3. Wire the data output from the DAQ Assistant **to the Signals input on the** Write to Measurement File Express VI.

5. Click **Run** to start the VI. After some time, press the **Stop** button on the Front Panel.

6. Your file will be created in the folder specified. If you can't remember the location of the file, right-click the **Write to Measurement File Express VI** and select **Properties**.

7. Using Windows Explorer, navigate to the location of the data file on disk.



Time	Voltage
4/21/2024 20:31:33.606	-0.29938
4/21/2024 20:31:33.606	-0.12329
4/21/2024 20:31:33.606	0.172732
4/21/2024 20:31:33.606	0.092775
4/21/2024 20:31:33.606	0.437324
4/21/2024 20:31:33.606	0.108949
4/21/2024 20:31:33.606	0.491646
4/21/2024 20:31:33.606	0.308843
4/21/2024 20:31:33.606	0.242008
4/21/2024 20:31:33.606	0.336309
4/21/2024 20:31:33.606	0.902113
4/21/2024 20:31:33.606	0.441291
4/21/2024 20:31:33.606	0.435798
4/21/2024 20:31:33.606	0.810254
4/21/2024 20:31:33.606	0.9131
4/21/2024 20:31:33.606	0.712291
4/21/2024 20:31:33.606	0.943923
4/21/2024 20:31:33.606	0.768139
4/21/2024 20:31:33.607	1.264363
4/21/2024 20:31:33.607	1.16701
4/21/2024 20:31:33.607	1.441062
4/21/2024 20:31:33.607	1.549401
4/21/2024 20:31:33.607	1.359884
4/21/2024 20:31:33.607	1.373922
4/21/2024 20:31:33.607	1.67422
4/21/2024 20:31:33.607	1.684901
4/21/2024 20:31:33.607	1.875334
4/21/2024 20:31:33.607	1.704433
4/21/2024 20:31:33.607	1.557336
4/21/2024 20:31:33.607	1.961395
4/21/2024 20:31:33.607	1.742885
4/21/2024 20:31:33.607	1.983673
4/21/2024 20:31:33.607	1.642176
4/21/2024 20:31:33.607	1.862211
4/21/2024 20:31:33.607	2.108797
4/21/2024 20:31:33.607	2.318456
4/21/2024 20:31:33.607	2.166171
4/21/2024 20:31:33.607	2.380713

Figure 4. The resulting voltage data file from our LabVIEW application.

8. Close the data file.

9. You can also place a File Path control on your Front Panel. Right-click on the Front Panel navigate to **Fuse Design System >> String & Path** and select **File Path Control**. Place this on the Front Panel.

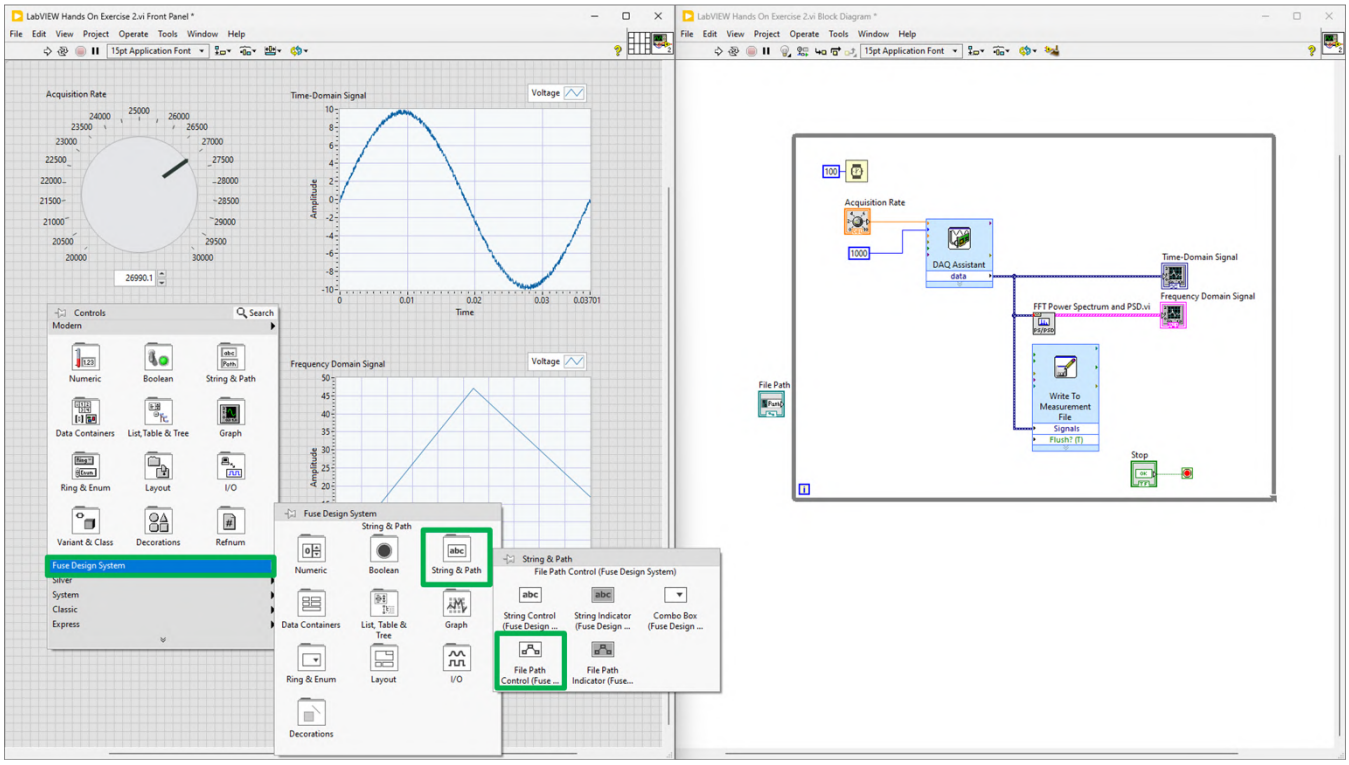


Figure 5. Add a File Path Control **to the Front Panel.**

10. You might want to resize the object and move it above the **Stop** button.

11. Move this inside the while loop and connect it to the Filename terminal in the **DAQ Assistant**.

12. Click **Run** to start the VI. After some time, press the **Stop** button on the Front Panel.

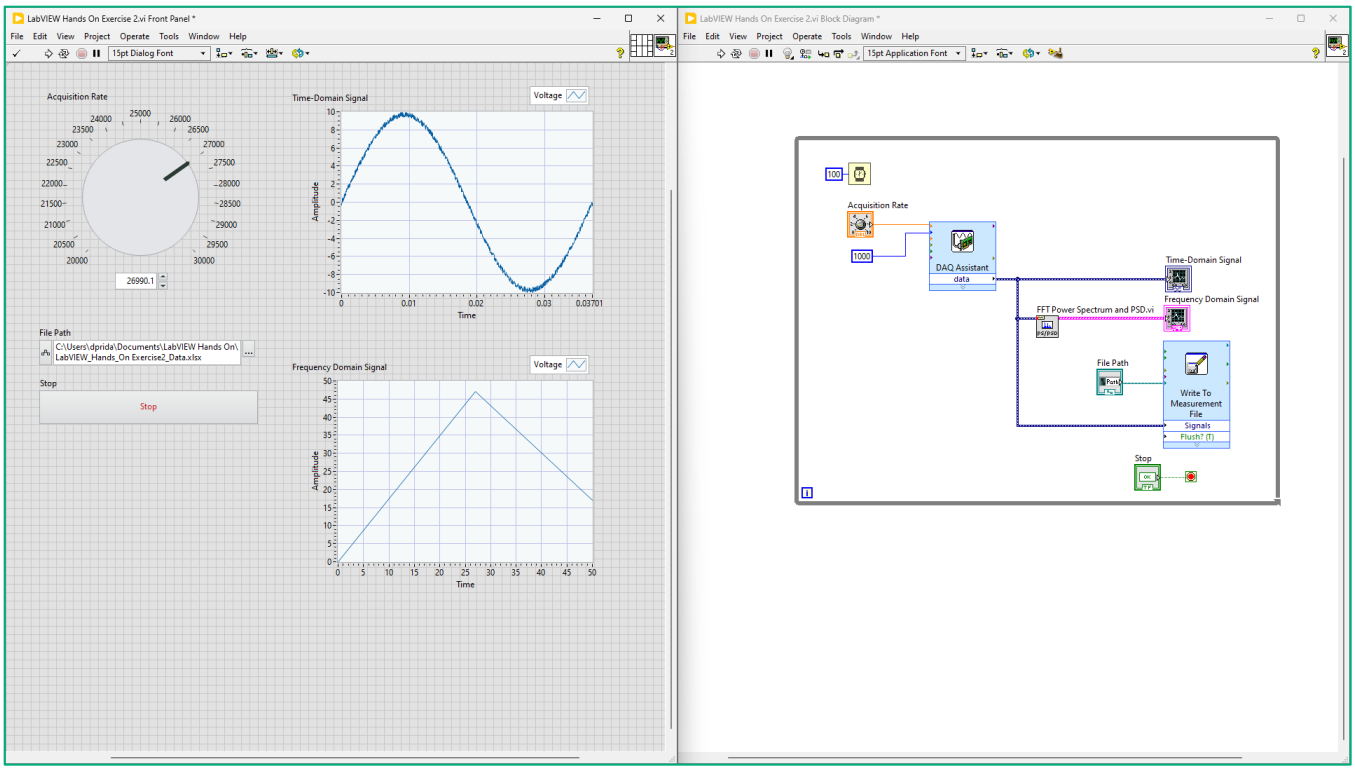


Figure 6. The File Path **control will make it easier to update the save location, without needing to open the** Write to Measurement File **Properties.**

Exercise Key Concepts:

In this exercise, we converted our first application to a more practical example – a voltage acquisition program. The DAQ Assistant and Write to Measurement File Express VIs provided a quick, configuration-based approach to aid in our development. These built-in functions help save a lot of time.

<End of Exercise>

Exercise 3: Acquire Data from Two Devices

Estimated time: 65 minutes.

Goals:

- Acquire data using the NI-DAQmx API.

Part A: Use the NI-DAQmx API

Estimated time: 20 minutes.

In Part A, you will replace the DAQ Assistant with the more flexible lower-level NI-DAQmx VIs. As the name suggests, Express VIs provide a fast and easy way of programming in LabVIEW, however, the DAQ Assistant isn't quite as flexible or scalable for mixed-measurement applications.

1. Delete the **DAQ Assistant** in the Block Diagram.

2. Also, delete the **Acquisition Rate** knob and the **1000** Number of Samples constant.

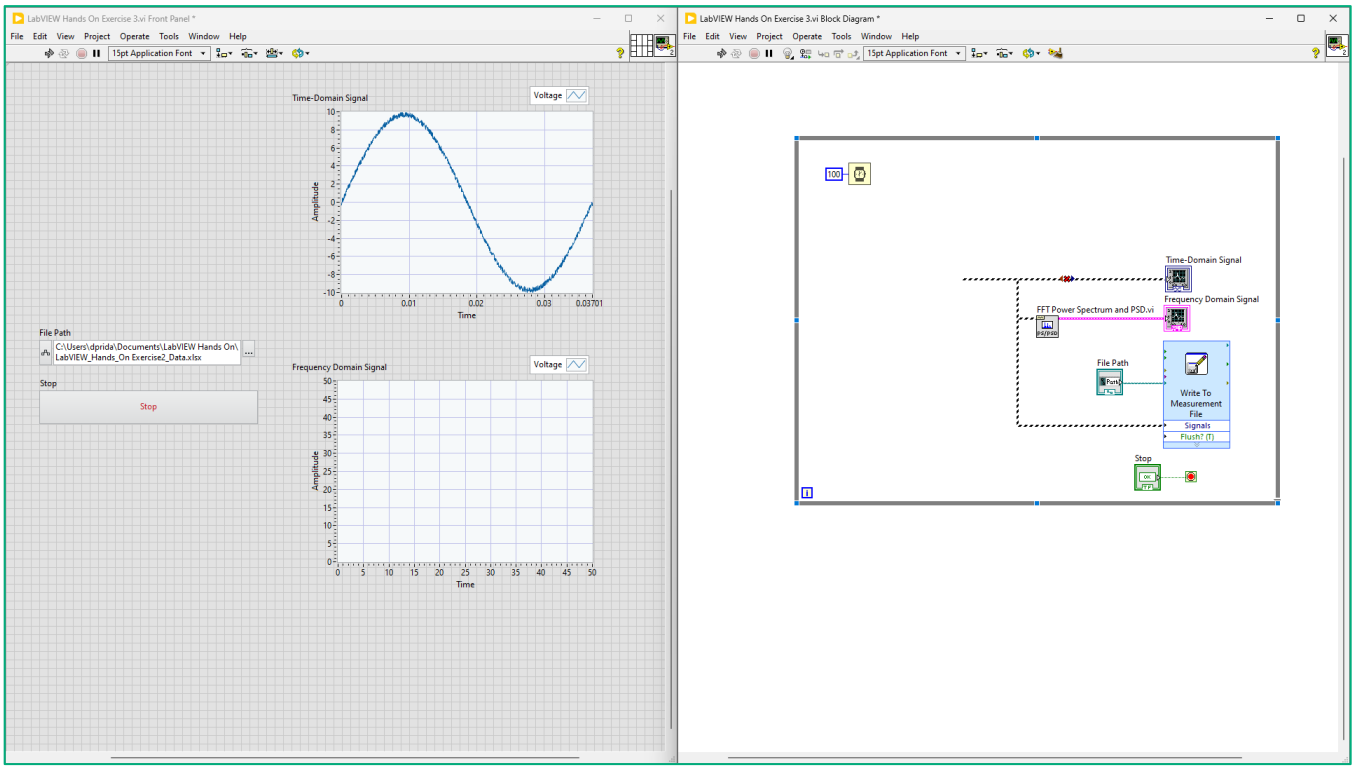


Figure 1. Delete the DAQ Assistant **and the accompanying** Acquisition Rate **and** Number of Samples.

3. Right-click on the Block Diagram. Navigate to **Measurement I/O >> NI DAQmx** and click on **Create Channel**. Place this outside, to the left of the **While Loop.**

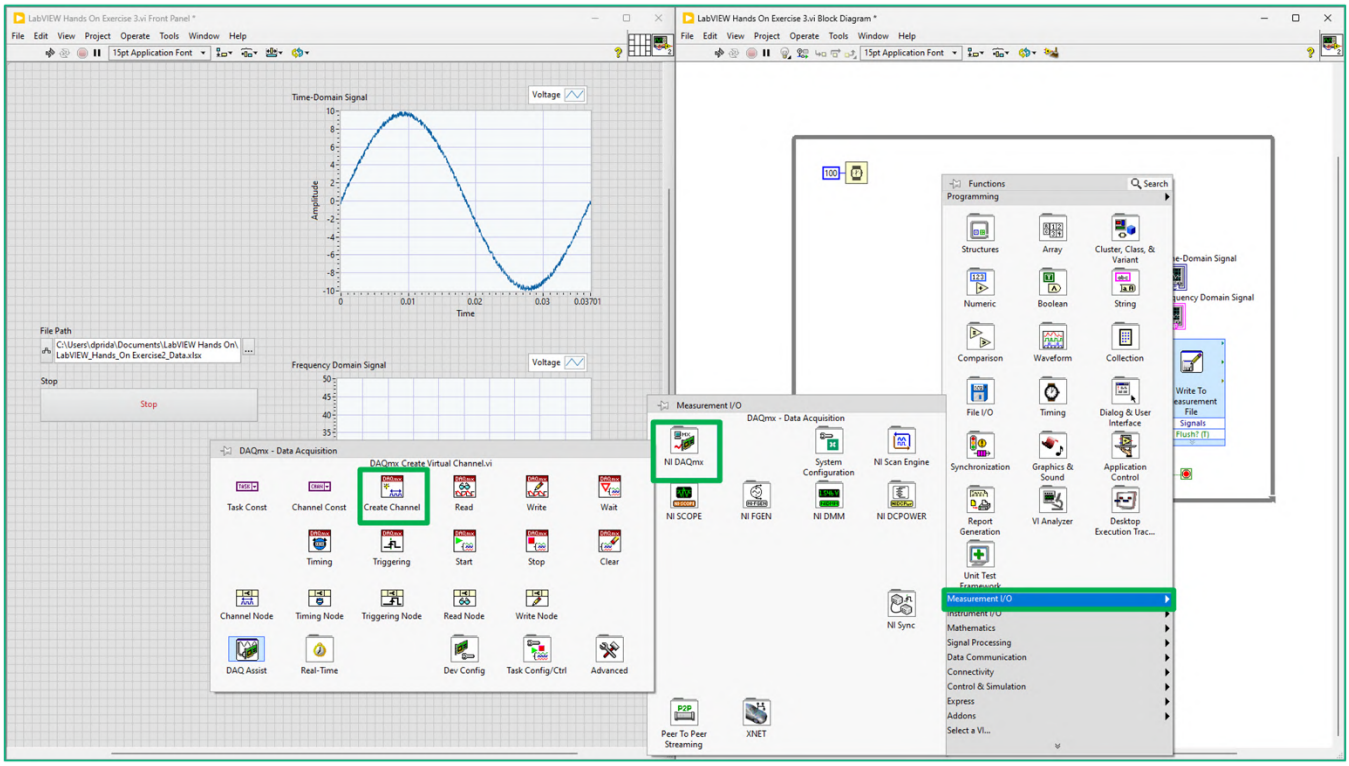


Figure 2. The NI-DAQmx palette houses all the data acquisition functions built into the LabVIEW environment by the NI-DAQmx driver.

LabVIEW Tip:

If you click on the push pin in the Controls or Functions Palette, the pinned palette will stay open after you add an item. Clicking the pin next to the DAQmx subpalette will keep that portion open. Unpin to close it.

4. Add the timing configuration. Right-click on the Block Diagram, navigate to **Measurement I/O » NI-**DAQmx, and select **DAQmx Timing**. Place this item outside of the **While Loop**.

5. Add the start block. Right-click on the Block Diagram, navigate to **Measurement I/O » NI-DAQmx**, and select **Start**. Place this item outside of the **While Loop**.

6. Add the data acquisition. Right-click on the Block Diagram, navigate to **Measurement I/O » NI-DAQmx**, and select **Read**. Place this item inside the **While Loop**. As we are acquiring data, we select Read. If we were generating a signal, we would select Write.

7. Add the stop and clear blocks. You will place both items outside of the **While Loop**, to the right. Right-click on the Block Diagram, navigate to **Measurement I/O » NI-DAQmx,** and select **Stop**. Place it. Right-click on the Block Diagram, navigate to **Measurement I/O » NI-DAQmx,** and select **Clear**. Place it.

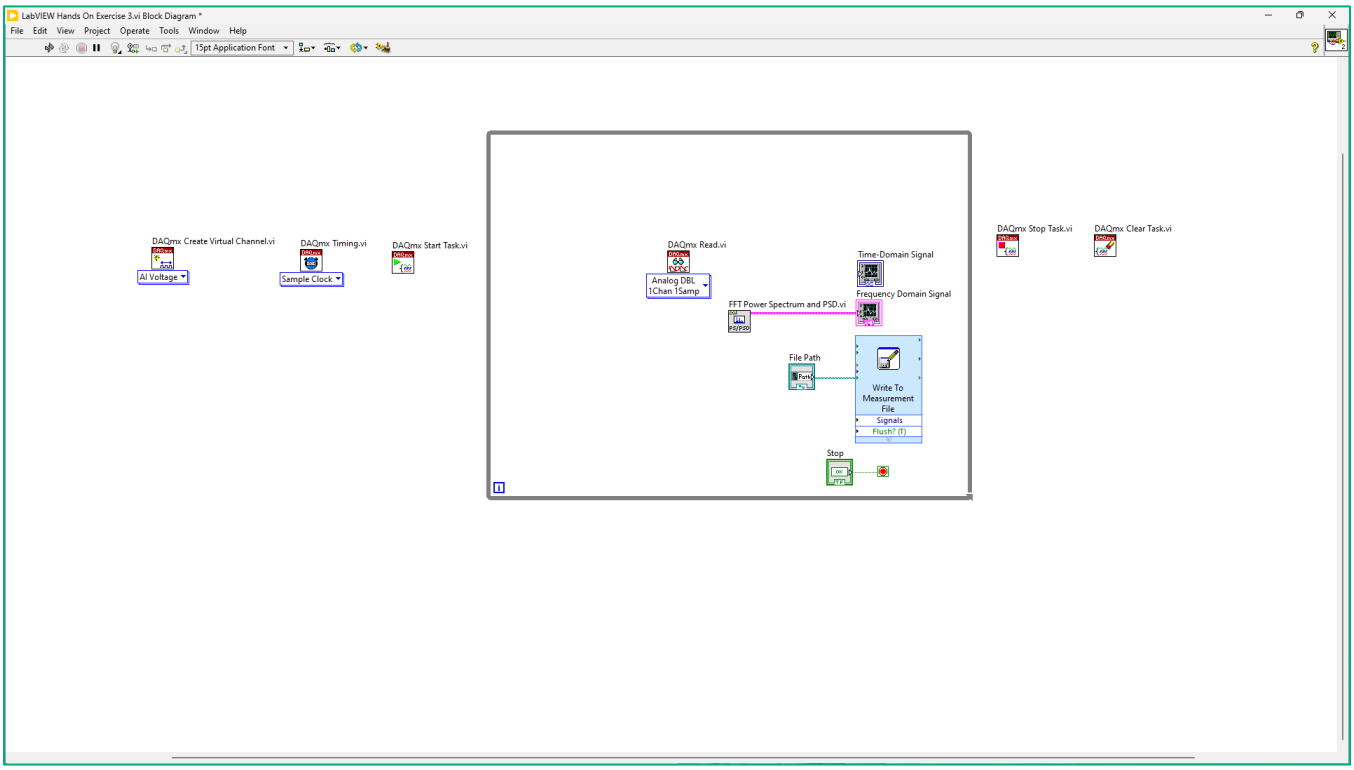


Figure 3. The Block Diagram should look like this after adding the DAQmx Create Channel, Timing, Start, Read, Stop, and Clear subVIs. This is a standard pattern for acquiring data.

Exercise Note:

Notice that the logical flow of the NI-DAQmx API begins by configuring an analog input channel that corresponds to a physical channel of measurement. Just before the while loop, the acquisition is started. Within the loop, samples are read from the NI-DAQmx buffer repeatedly until the user stops the loop, at which point the acquisition setup is cleared from memory.

8. Wire the task terminal output of the **DAQmx Create Virtual Channel** subVI, Task/Channels In, to the **DAQmx Timing** subVI.

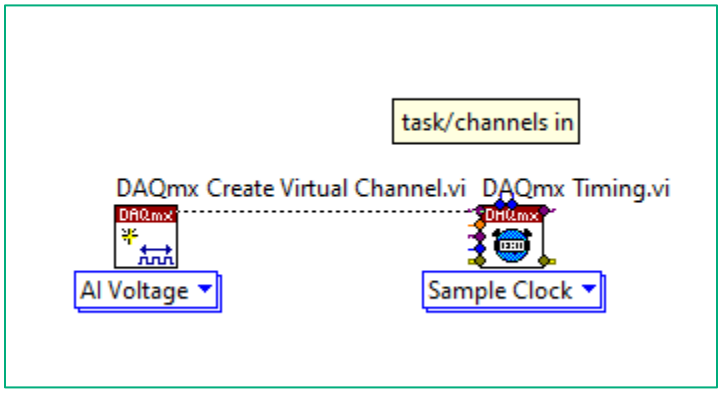


Figure 4. Wire the Task setting from the DAQmx Create Virtual Channel **to the** DAQmx Timing.

a. Wire and connect the task information for all the DAQmx subVIs.

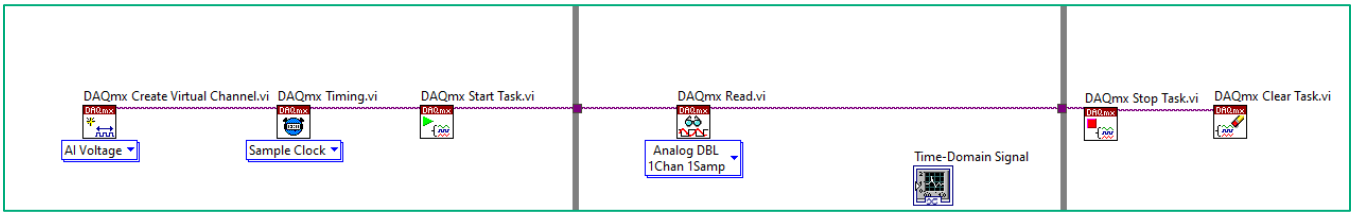


Figure 5. Wire and connect the task information for all the DAQmx subVIs.

9. Next, add error handling to the application. Without a mechanism to check for errors, you can only know that a VI does not work properly. Wire the bottom output of the **DAQmx Create Virtual Channel** to the **DAQmx Timing**

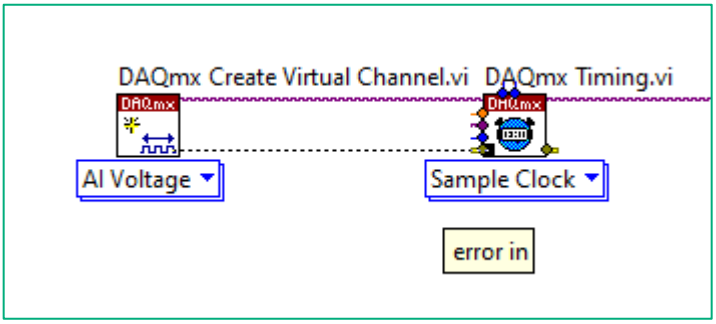


Figure 6. Wire and connect the error information for all the DAQmx subVIs.

a. Wire and connect the error information for all the DAQmx blocks.

10. Go to the Front Panel. At this point, all of the DAQmx functions have been added to the Block Diagram. Some settings will need to be configured…but first, we will add a control to select the channel we will be acquiring from. Right-click and navigate to **Fuse Design System >> I/O >> Fuse Design System DAQmx Name Controls** and select **DAQ Physical …**

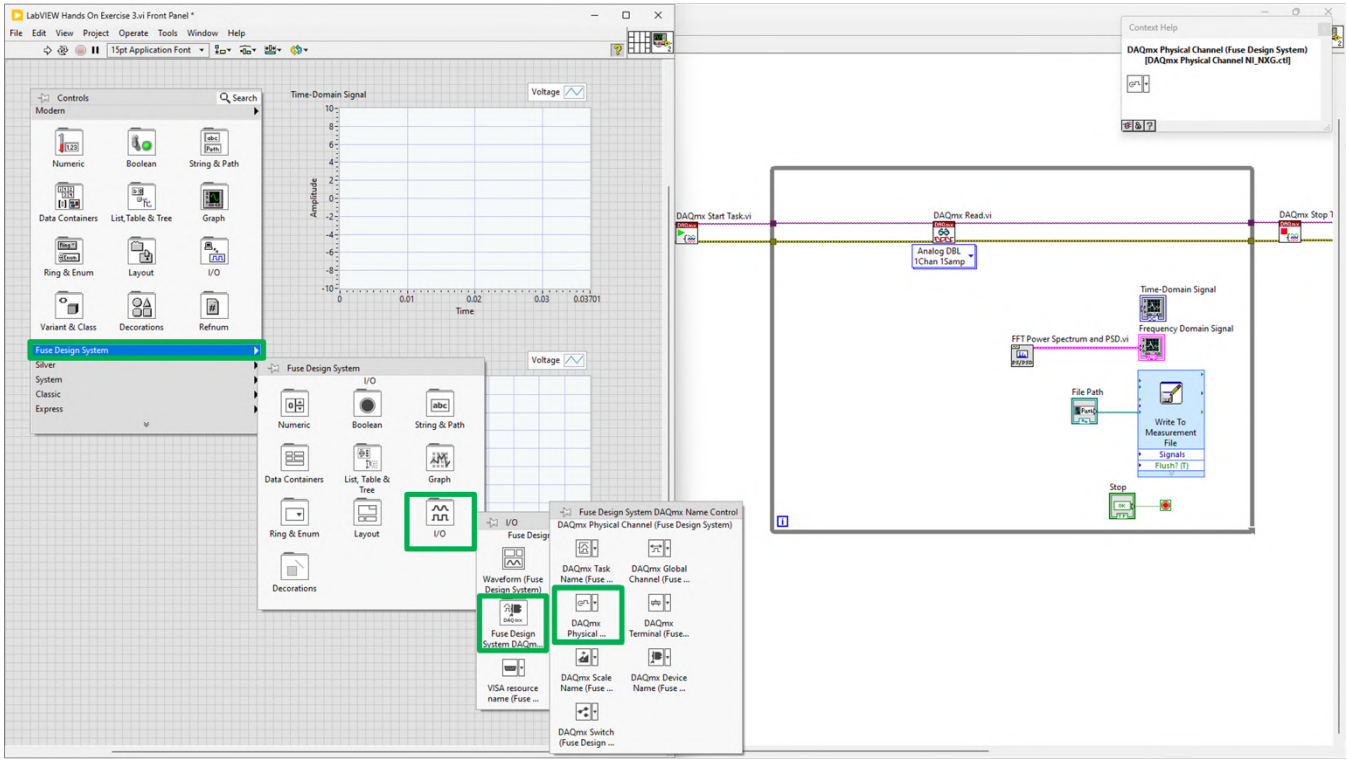


Figure 7. Add the DAQmx Physical Channel **control to the Front Panel**

a. Double-click the label **"DAQmx Physical Channel"** and type in **"Voltage Input".**

b. On the Block Diagram, wire the output of the **Voltage Input** to the second terminal input on the **DAQmx Create Virtual Channel** subVI.

11. On the Block Diagram, go to the DAQmx Timing subVI.

    a. At the top edge of the block, right-click on the Sample Mode terminal, go to **Create,** and select **Constant.** This should be set to **Continuous Samples**. If not, use the dropdown to change it.

    b. Next to the Sample Mode terminal, right-click on the Number of Samples terminal, go to **Create,** and select **Constant.** This should be set to **1000**.

    c. On the left edge, go to the Rate input. Right-click, go to **Create,** and select **Constant**. Leave the default rate value of 1000 Hz.

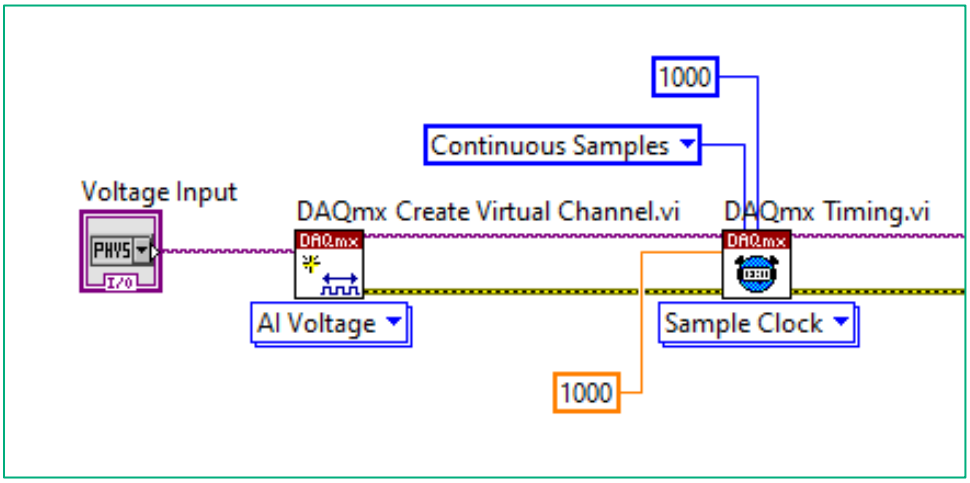


Figure 8. Complete the configuration of the DAQmx Timing **subVI as shown.**

12. Next, configure the DAQmx Read subVI. On the **DAQmx Read** subVI, click on the dropdown below the block. Go to **Analog >> Single Channel >> Multiple Samples** and select **Waveform (Samples)**.

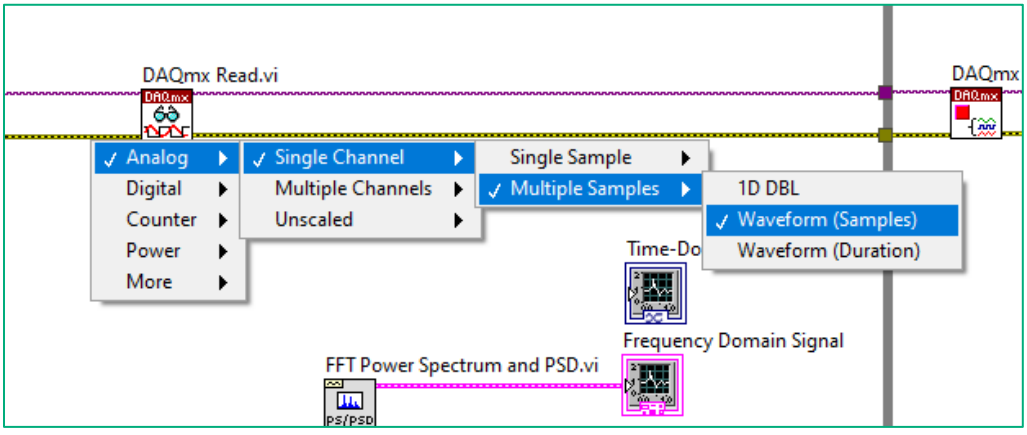


Figure 9. Configure the DAQmx Read **subVI.**

13. Wire the data output of the **DAQmx Read** subVI to the **Time-Domain Signal** graph, **FFT Power Spectrum and PSD** subVI, and the **Write to Measurement** File subVI.

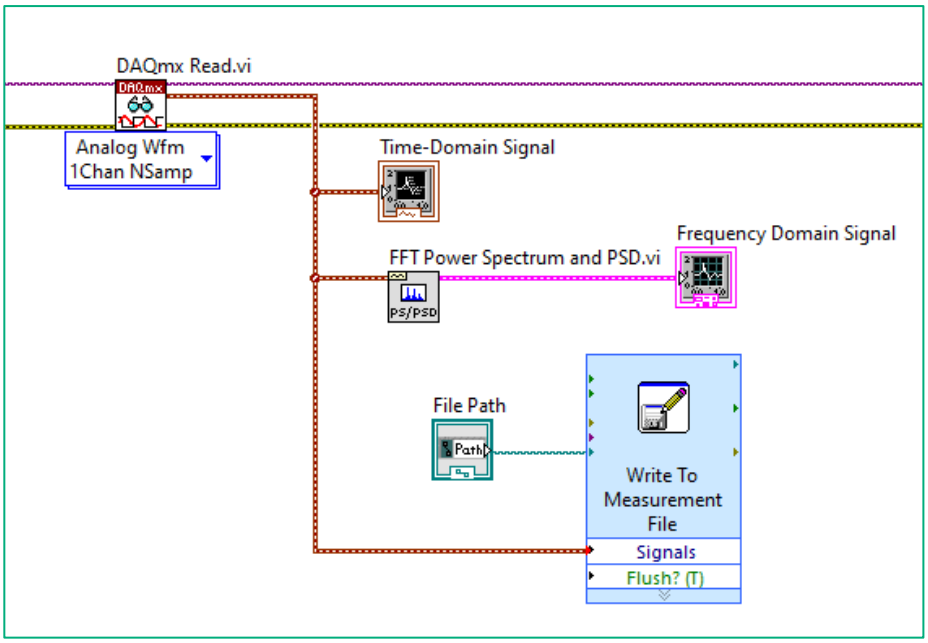


Figure 10. Wire the data output of the DAQmx Read **subVI.**

14. It might be worth spending a few moments to clean up the Block Diagram. Move blocks to eliminate overlaps and unnecessary wire branches. Resize the While Loop to eliminate blank space.

15. Go to the Front Panel. On the **Voltage Input** menu, click the dropdown and scroll to select **Voltage_in/ai0**.

16. On the Front Panel for the voltage Time Domain signal, a graph is currently being used. If we were to run the application, it would only show the data obtained from a single run of the while loop. We want to see the current data and the history so we will replace it with a chart. Right-click on the **Time-Domain Signal** graph. Navigate to **Replace**. The Controls Palette will open. Navigate to **Fuse Design System >> Graph** and select **Waveform Chart.**

    a. Right-click on the **Y-axis** to turn off AutoScaling. Double-click on the Y-axis minimum and maximum and type to set it to **-5** and **5**, respectively.

13. Click **Run** to start the VI. After some time, press the **Stop** button on the Front Panel.

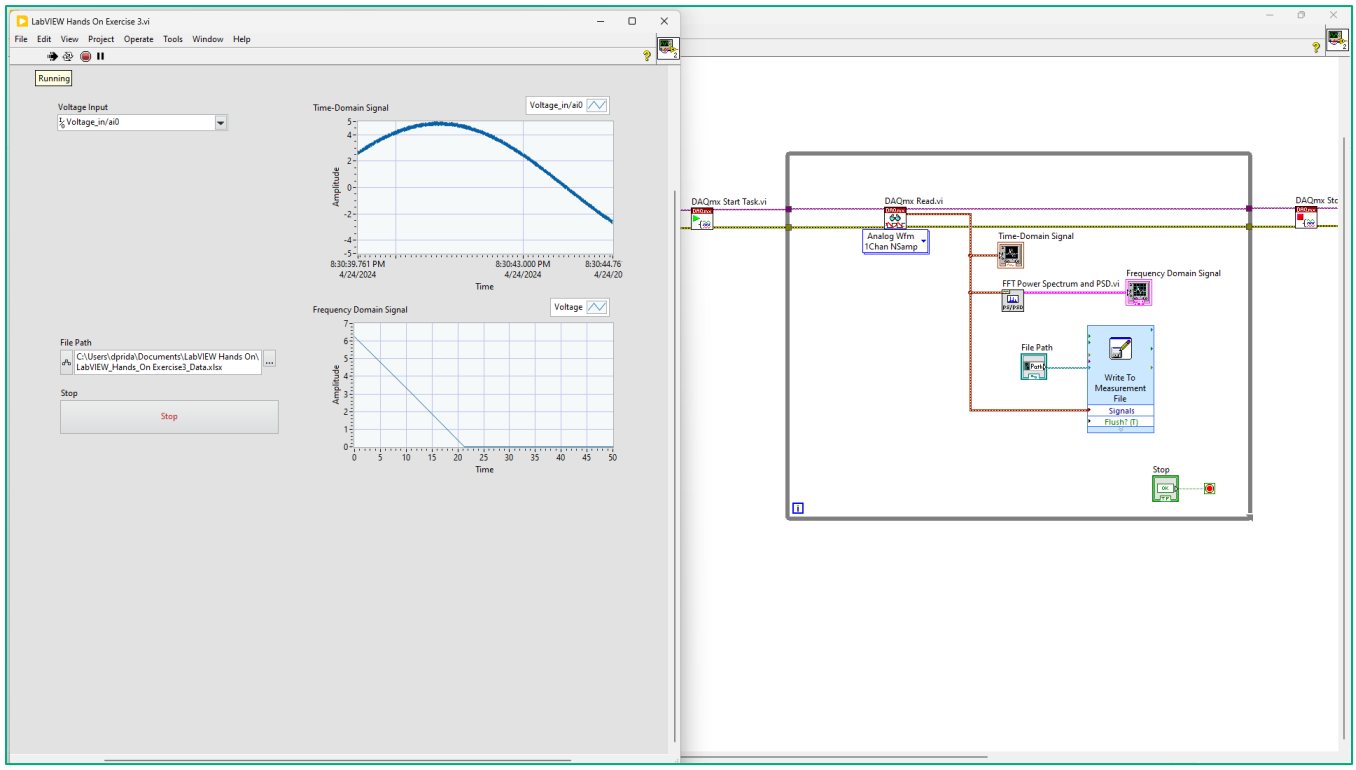


Figure 11. The complete application should look like this.

Exercise Note:

A Waveform Graph plots all the received points at once. A Waveform Chart displays received data in addition to already existing points.

Part B: Add a Temperature Measurement

Estimated time: 20 minutes.

In Part B, you will add another measurement to your application. LabVIEW can acquire data from any instrument (regardless of vendor) and multiple instruments at once - LabVIEW's inherent parallelism allows multitasking and multithreading without complex coding.

1. Go to the Block Diagram and type **Ctrl+A** to select all the code that we have created so far. Type **Ctrl+C** to copy and **Ctrl+V** to paste. While the new section is still highlighted after pasting, move the new section of code below. [If you make a mistake, you can try again with **Ctrl+Z**.]

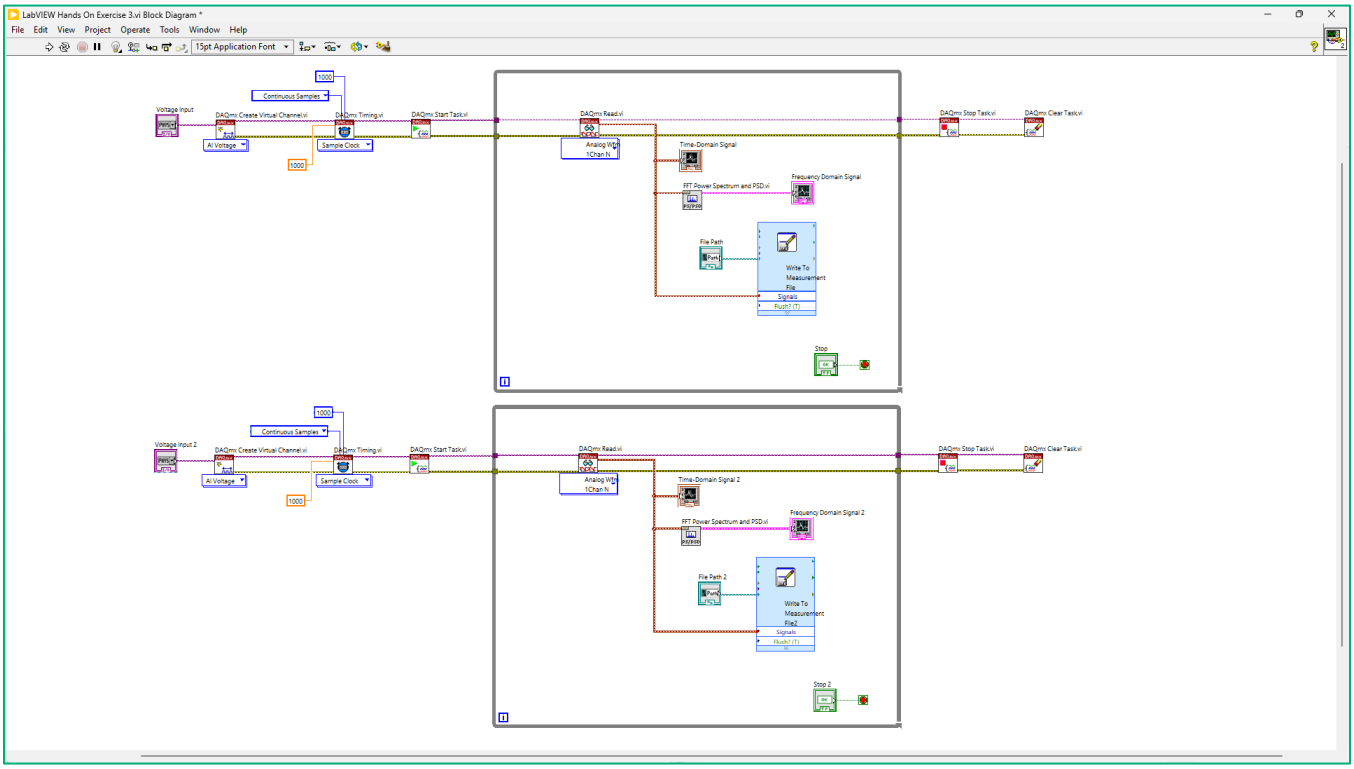


Figure 1. Copy and paste the code from the voltage measurement to use for the temperature measurement.

LabVIEW Tip:

You can zoom on the Block Diagram. If you need to better view or review sections of code. It can be helpful to take a closer look or take a step back. On the menubar, go to **View** and select **Toggle Zoom**. You can also use keyboard shortcuts, too: Zoom In: **Ctrl++**, Zoom out: **Ctrl+-**, and Actual Size: **Ctrl+0**

2. Delete the **Write to Measurement File** and the **FFT Power Spectrum**, **Frequency Domain Signal** graph and the **Stop 2** button attached to the Loop Condition. Type **Ctrl+B** to delete any broken wires.

    a. You can resize the While Loop to eliminate white space.

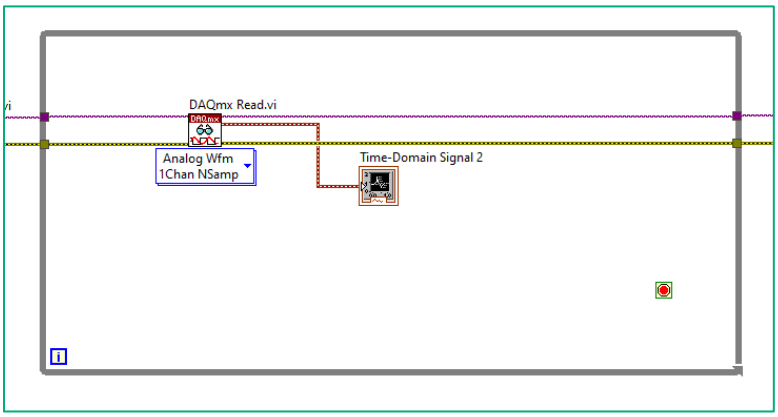


Figure 2. The While Loop for the Temperature Measurement should now contain the DAQmx Read, the Time Domain Signal 2, and the Loop Condition.

3. To better distinguish these two loops, we will add labels. Right-click on the top **While Loop**, navigate to **Visible Items,** and select **Subdiagram Label**. Type in **"Voltage Acquisition"**. Then, do the same for the bottom While Loop and label it as **"Temperature Acquisition"**.

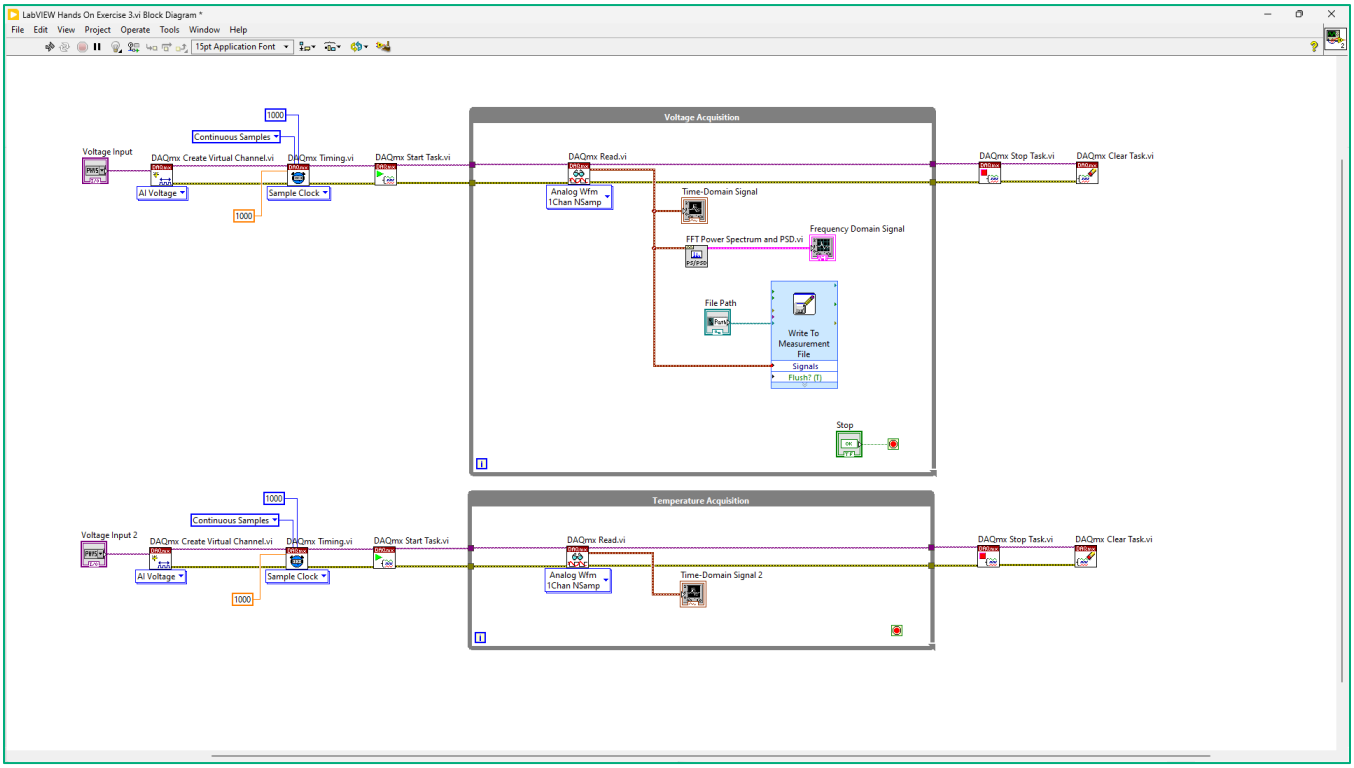


Figure 3. Adding labels and notes to your Block Diagram can help provide clarity.

4. Go to the Front Panel and organize the newly pasted objects. You may need to scroll to find the objects. If you're having trouble, you can go to the Block Diagram and double-click on the item's icon and it will highlight it on the Front Panel.

    a. Move the **Time Domain Signal 2** graph below the **Frequency Domain Signal**.

    b. Double-click the label **Time Domain Signal** and type in **Temperature (C).**

    c. Double-click the label **Frequency Domain Signal** and type in **Voltage FFT.**

    d. Double-click the label **Time-Domain Signal** and type in **Voltage (V)**.

    e. Double-click the label **Voltage Input 2** and type in **Temperature Input**.

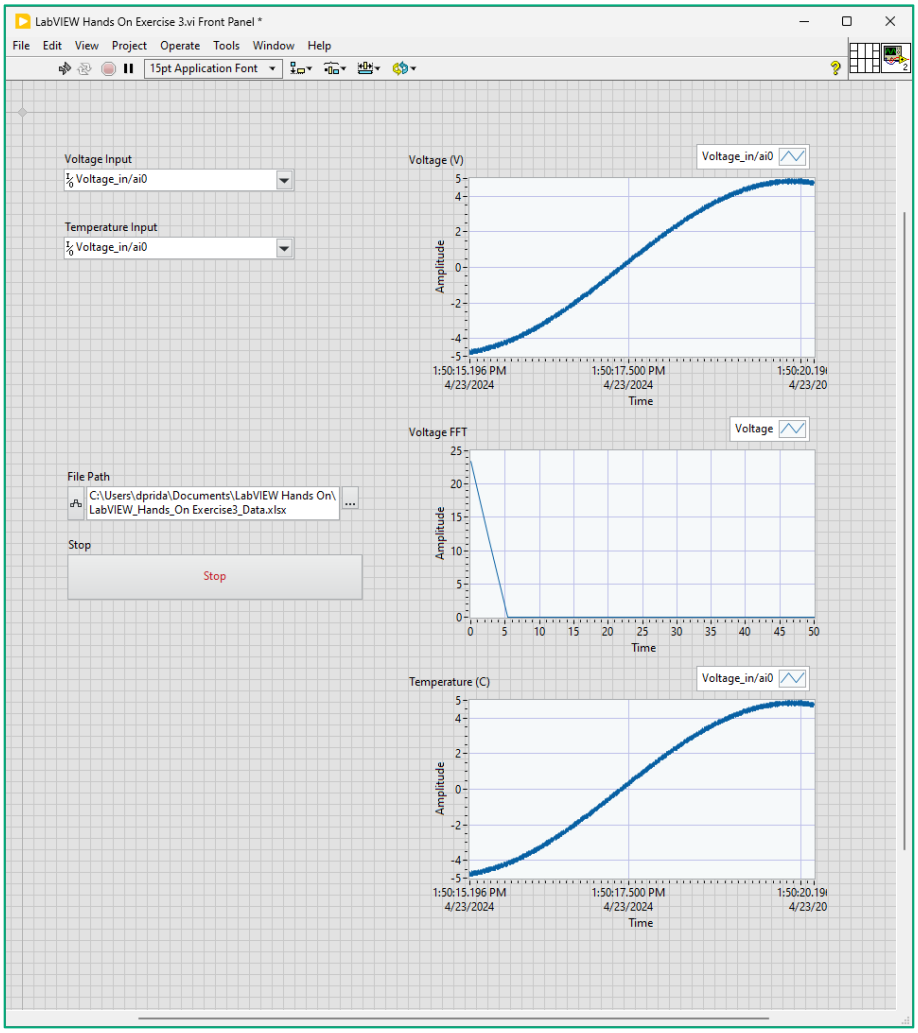


Figure 4. The Front Panel should look like this.

5. Go to the Block Diagram, on the second **DAQmx Create Channel** subVI, click to the drop-down menu. Navigate to **Analog Input >> Temperature** and select **Thermocouple**.

6. On the **DAQmx Timing** subVI change the Rate to **10** and the Number of Samples to **10**.

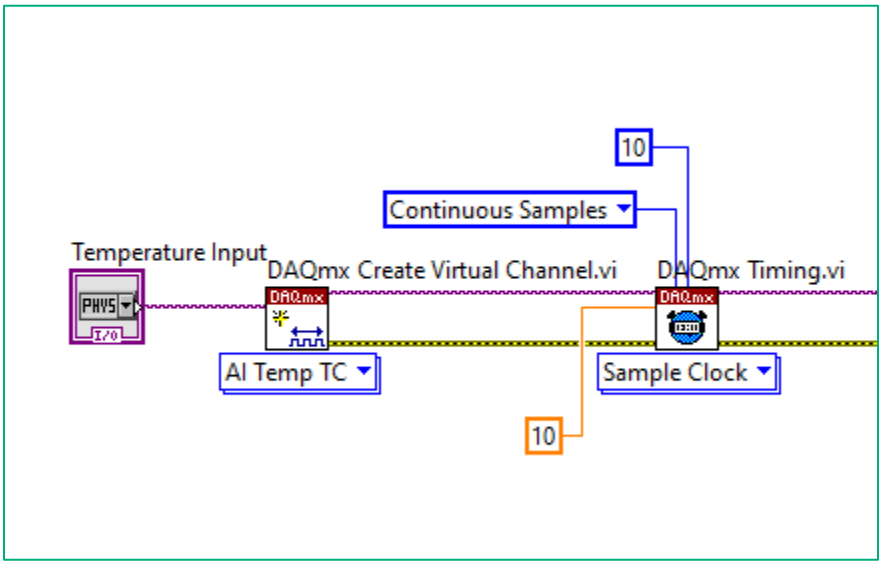


Figure 5. Configure the DAQmx Create Virtual Channel **and** Timing **subVI as shown.**

7. Currently, there are two continuous processes in our application with the two While Loops. We need to be able to control and stop the application. For this, we will be using a quick approach with a Local Variable. Right-click on the Block Diagram, navigate to **Data Communication** and select **Local Variable**. Place this inside the lower While Loop.

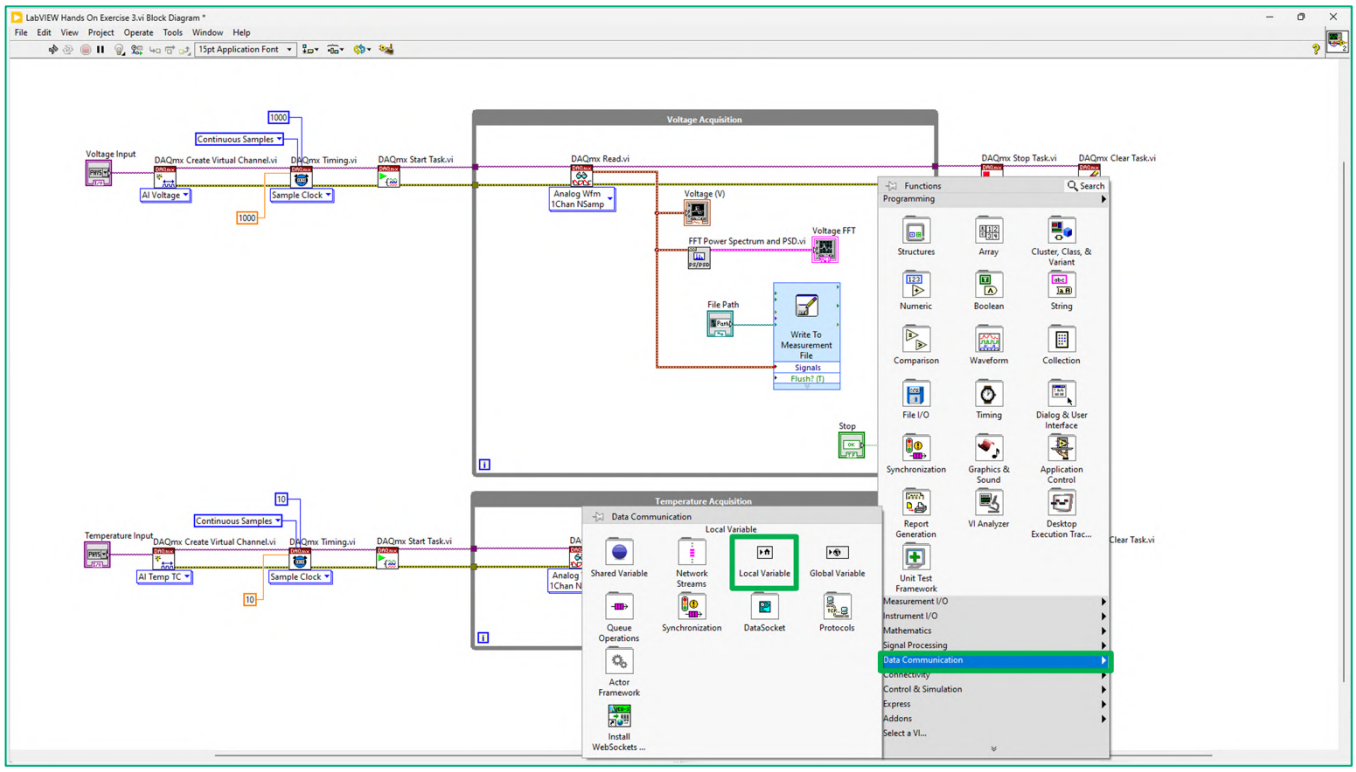


Figure 6. Add a Local Variable **to the Temperature Acquisition While Loop.**

a. Click on the **Local Variable** and a dropdown menu will appear. Select **Stop**. This will read the stop information from the Voltage Acquisition While Loop.

b. Right-click on the **Stop** variable and click **Change to Read**

c. Wire the connect the **Stop** variable to the **Loop Condition**.

Exercise Note:

There are several ways to stop two parallel While Loops in LabVIEW. We have selected using a Local Variable as it's a quick and easy method, however, it's not necessarily a best practice. There are more robust solutions for more advanced applications, such as a Notifier or Queue.

8. On the Front Panel, change the Mechanical Action of the Stop button. Right-click on the **Stop** button. Navigate to **Mechanical Action** and select **Switch When Released**.

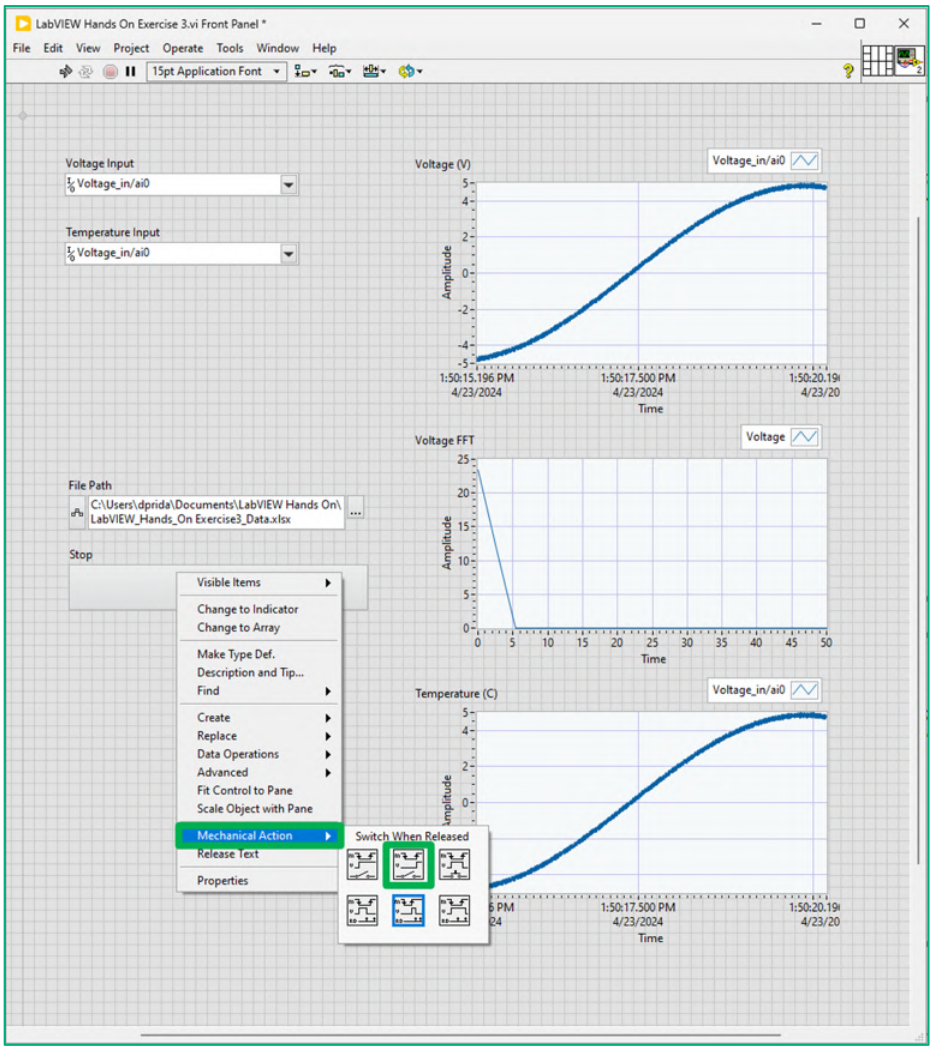


Figure 7. Boolean controls have six types of mechanical action allowing you to customize the objects to resemble the behavior of physical instruments. Latch actions cannot be used for objects with a local variable as race conditions can occur. (You will notice the broken Run arrow preventing you from starting the application.) Set the Stop **button to** Switch When Released.

9. Adjust the Temperature graph. Right-click on the Y-axis of the **Temperature graph** and click on **Autoscaling** to turn it off.

    a. Double-click on the minimum and maximum values to and type in **-100** and **100**, respectively.

10. Click on the dropdown menu of the **Temperature Input** and select **Temperature/ai0**.

11. Click **Run** to start the VI. After some time, press the **Stop** button on the Front Panel.

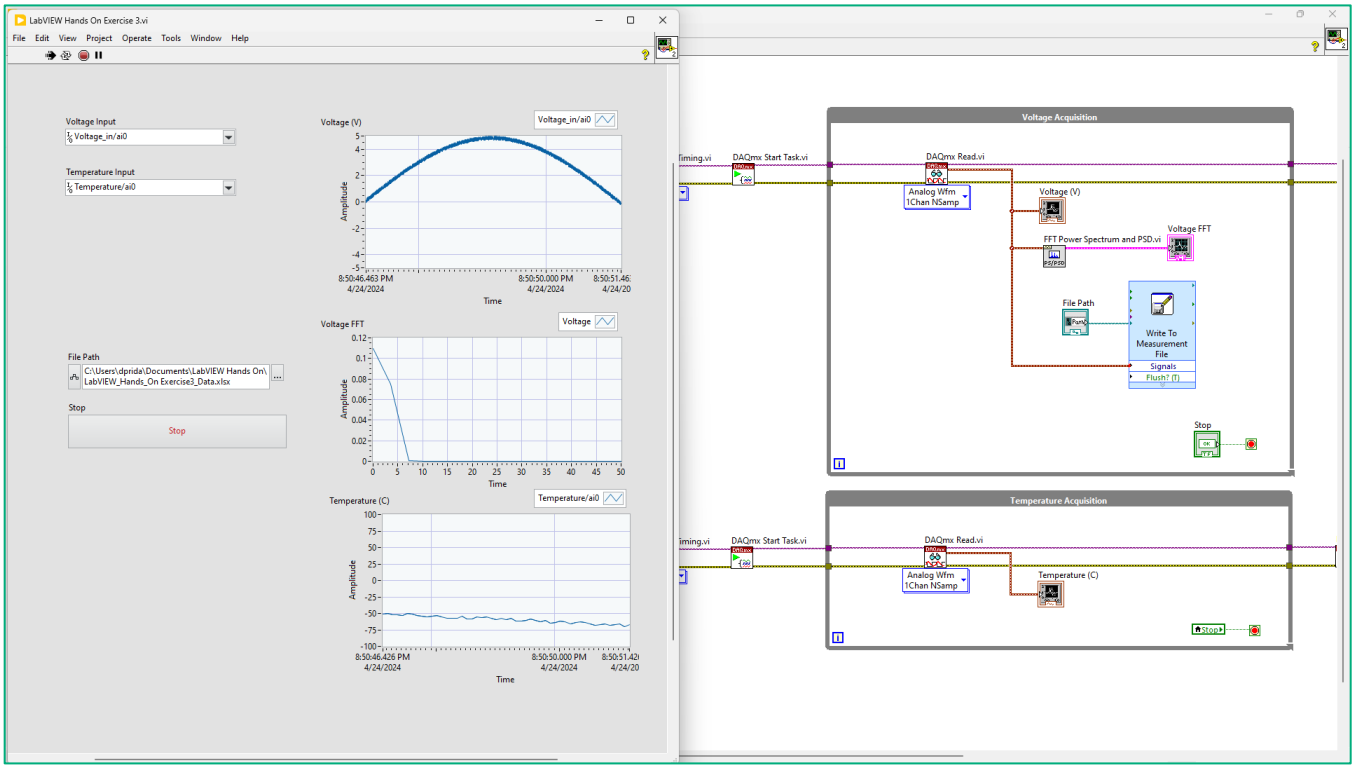


Figure 8. The completed application will acquire the voltage and temperature data simultaneously.

Part C: Add a Notification

Estimated time: 10 minutes.

In Part C, you will add an alert to notify the user if the temperature has exceeded a limit.

1. Right-click on the Block Diagram. Navigate to **Programming >> Waveform >> Analog Waveform** and select **Min Max**. Place this inside the Temperature Acquisition While Loop.

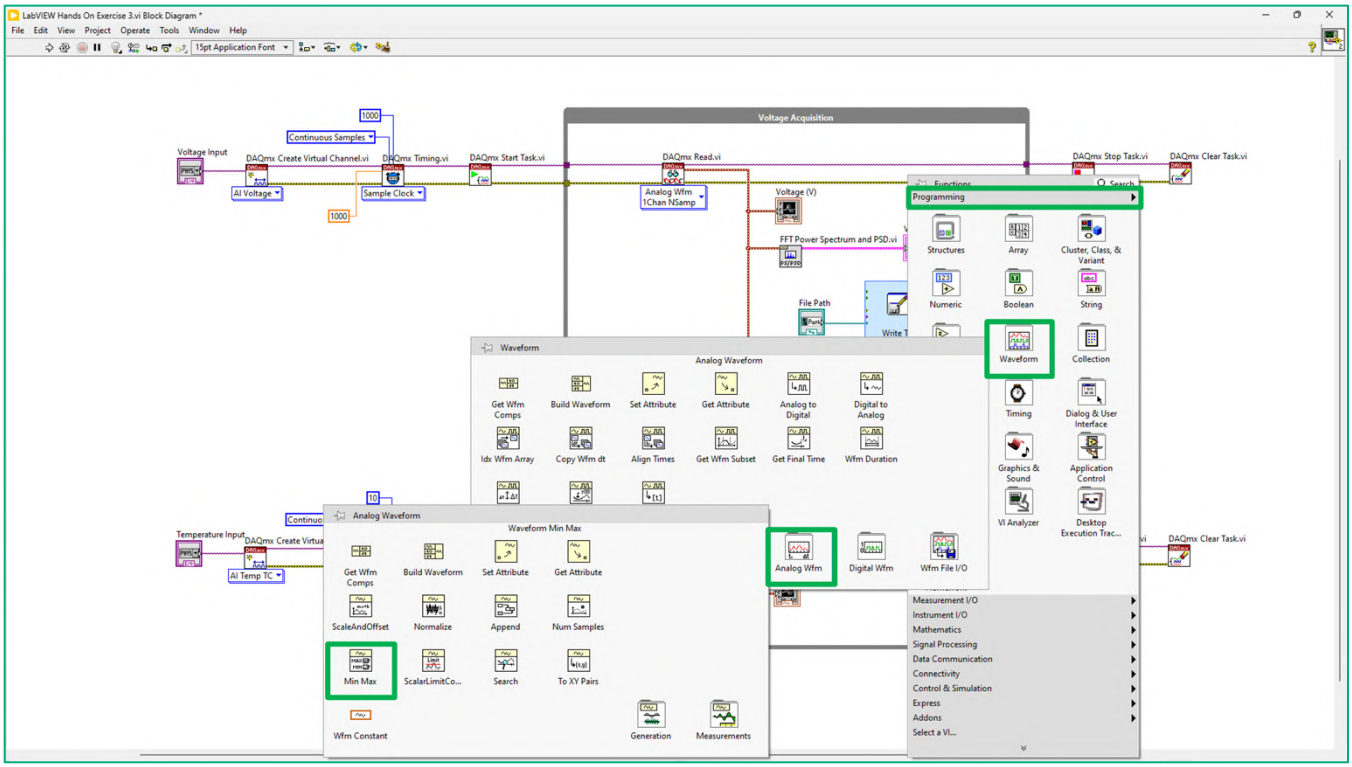


Figure 1. The configuration dialog of the Comparison Express VI.

a. Right-click on the **Waveform Min Max** subVI. Navigate to **Visible Items** and select **Label**.

b. Wire the waveform input on the **Waveform Min Max** subVI to the data output of the **DAQmx Read** subVI.

2. To check if the limit has been exceeded, a comparison function needs to be added. Right-click on the Block Diagram, navigate to **Programming >> Comparison** and select **Greater**. Place this inside the Temperature Acquisition While loop.

    a. Wire the one input of the **Greater** function to the data output of the **DAQmx Read** subVI.

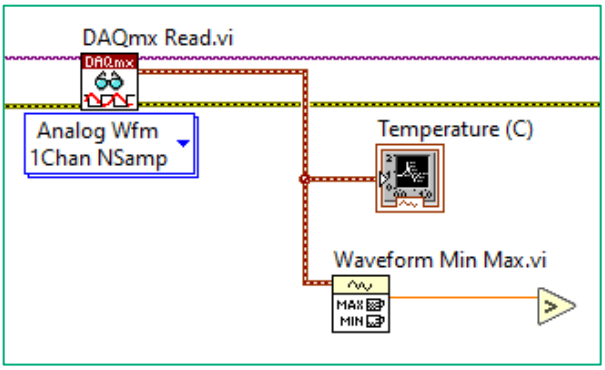


Figure 2. Wire the Waveform Min Max **subVI as shown.**

3. Add a **Wait** function back to the Block Diagram in the **Temperature Acquisition** While Loop.

4. Go to the Front Panel. Right-click and navigate to **Fuse Design System >> Numeric** and select **Horizontal Slider**. Place this on the Front Panel below the Temperature Input.

    a. Double-click on the **"Horizontal Slider"** and rename it to **"Temperature Limit"**.

    b. Use the resizing handles as needed.

5. Right-click and navigate to **Fuse Design System >> Boolean** and select **LED.**

    a. Double-click on the **"LED"** and rename it to **"Warning!"**.

    b. Use the resizing handles as needed.

6. Wire the **Temperature Limit** and **Warning!** appropriately on the Block Diagram. This is the final step of programming in our application.

7. Click **Run** to start the VI. After some time, press the **Stop** button on the Front Panel.

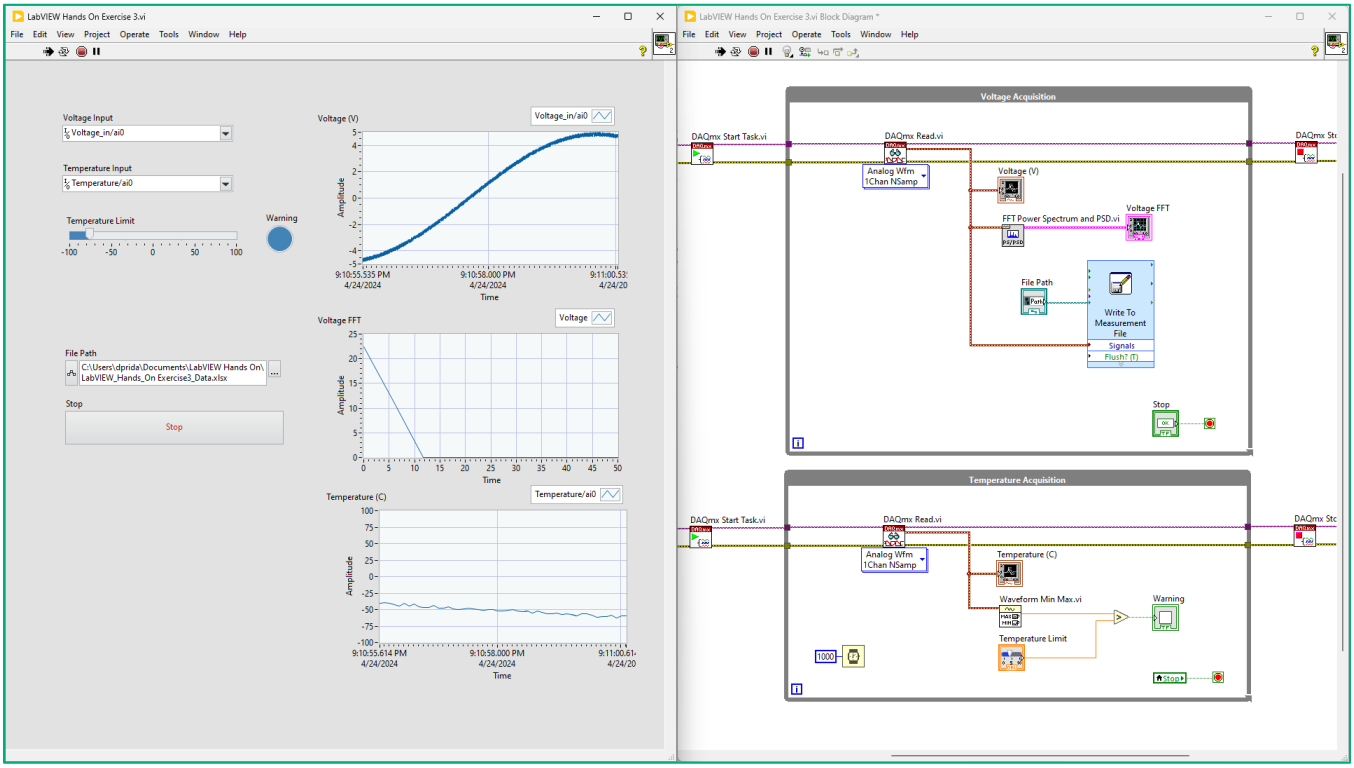


Figure 3. The completed application should look like this.

Exercise Key Concepts:

In this exercise, we expanded our second application to include another measurement. LabVIEW can control and interact with all your instruments. And with parallel processing, it can do so simultaneously. LabVIEW's inherent parallelism made the process straightforward; we didn't need advanced skills.

Bonus: Improve the User Interface Design

Estimated time: 20 minutes.

You have completed your application and it's completely functional. Depending on the user, you might want to spend some more time improving the user interface to make it more intuitive.

1. At the top of your Front Panel, double-click to start typing a free label. Type in **"LabVIEW Hands-On"**. Click anywhere on the Front Panel to finish.

    a. As you hover over the text item, you can select and resize it. **Click** on the object to select it.

    b. On the Menu bar go to the **Text Settings**. Here, you can change the font, style, size, and color. Go to **Size** and select **36.**

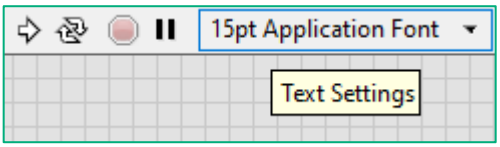


Figure 1. The Text Settings can be used to modify the size, color, and font.

2. Create a section label for the **Voltage** and **Temperature Input**. Double-click to start typing a free label. Type in **"Hardware Settings"**. Click anywhere on the Front Panel to finish. Move this below the application title but above the **Voltage Input**.

    a. Go to the **Text Setting >> Size** and select **24.**

    Go to the **Text Setting >> Style** and select **Bold.**

3. Create a section label for the **Temperature Limit** and **Warning**. Double-click to start typing a free label. Type in **"Temperature Warning"**. Click anywhere on the Front Panel to finish. Move this below **Voltage Input** but above the **Temperature Limit** slider.

    a. Go to the **Text Setting >> Size** and select **24.**

    b. Go to the **Text Setting >> Style** and select **Bold.**

4. Create a section label for the **File Path**. Double-click to start typing a free label. Type in **"File Save Location"**. Click anywhere on the Front Panel to finish. Move this below **Temperature Limit** but above the **File Path**.

    a. Go to the **Text Setting >> Size** and select **24.**

    b. Go to the **Text Setting >> Style** and select **Bold.**

5. Separate the sections by adding a line. Right-click on the Front Panel, navigate to **Fuse Design System >> Decorations**, and select **Horizontal Line**. Place

6. Right-click on the Front Panel, navigate to **Fuse Design System >> Decorations**, and select **Flat Box**. Place this in the top left corner of your application.

    a. Use the resizing handles to cover most of the Front Panel screen.

7. While the Flat Box is still selected, go to the menubar and find the Reorder icon. Click on **Reorder** and select **Move to Back**.

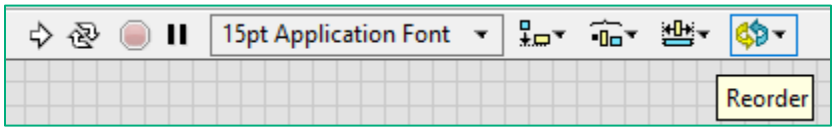


Figure 2. The Reorder button can be used to rearrange the stacking of items on the Front Panel.

8. While the Flat Box is still selected, go to the menubar and find the Reorder icon. Click on **Reorder** and select **Move to Back**.

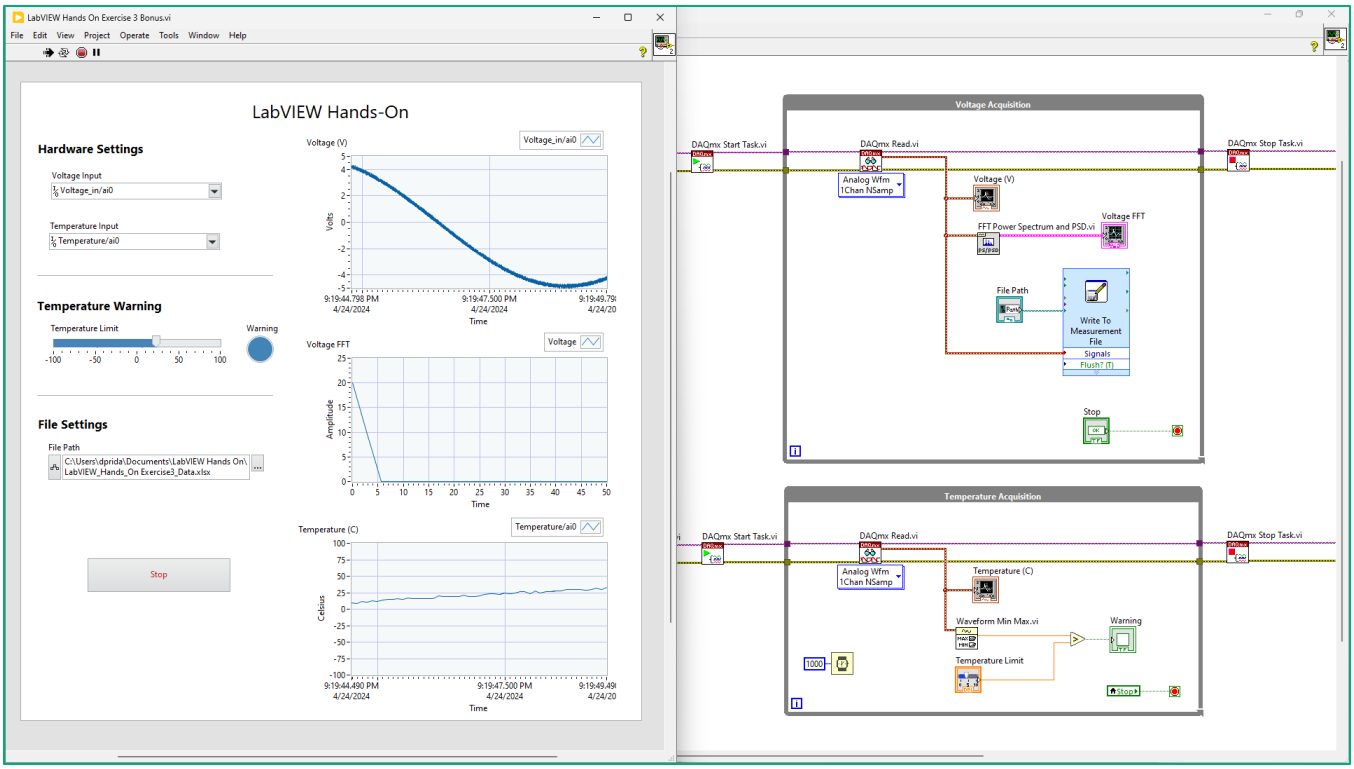


Figure 3. The Front Panel and Block Diagram are the same used on the cover of this manual.

<End of Exercise>

Exercise 4: Running a Data Acquisition Example

Goals:

- Become familiar with the Example Finder and NI-DAQmx examples.
- Run a pre-built example program to acquire your first measurement.

Part A: Open LabVIEW and Explore the Example Finder

Estimated time: 10 minutes.

LabVIEW device drivers, such as NI-DAQmx, often come with example programs. The NI-DAQmx LabVIEW Examples provide you with a proven starting point for creating data acquisition applications. By utilizing an example program, you can eliminate several sources of errors and save time by building on existing code.

1. You can open the example directory at any point while using LabVIEW. Go to the **Help** menu and select **Find Examples**.

2. Once the NI Example Finder launches, navigate to **Hardware Input and Output » DAQmx**.

3. Navigate the DAQmx directory and read some of the capabilities of the examples. Each example has a descriptive name, but you can find out more capabilities of the program by reading the descriptions in the Information section.

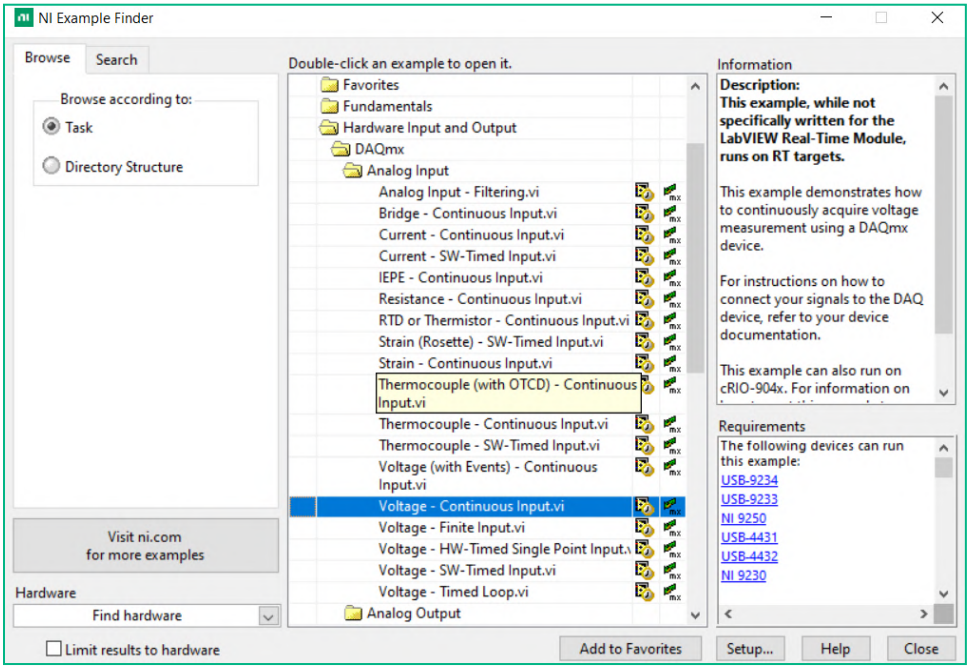


Figure 3. The NI Example Finder is a useful tool for navigating the installed NI examples.

Each section of the directory includes examples for performing certain tasks. For example, analog input includes examples for measuring simple voltage, current, strain, and temperature, whereas the analog output section includes examples for controlling either voltage or current.

Part B: Run a Pre-Built Example

Estimated time: 10 minutes.

Now that you are familiar with the Example Finder, you can open and run an example.

1. In the NI Example Finder, navigate to **Hardware Input and Output » DAQmx » Analog Input** and double-click **Voltage – Continuous Input.vi**. This will launch a VI to continuously measure a voltage channel.

2. When the VI launches, the Front Panel will be displayed. This is the user interface and includes controls and indicators for interacting with your code. In this example, notice the controls for selecting the channel, voltage input ranges, timing parameters and logging location. Additionally, the Front Panel includes a graph indicator for instantly viewing the data on the voltage channels.

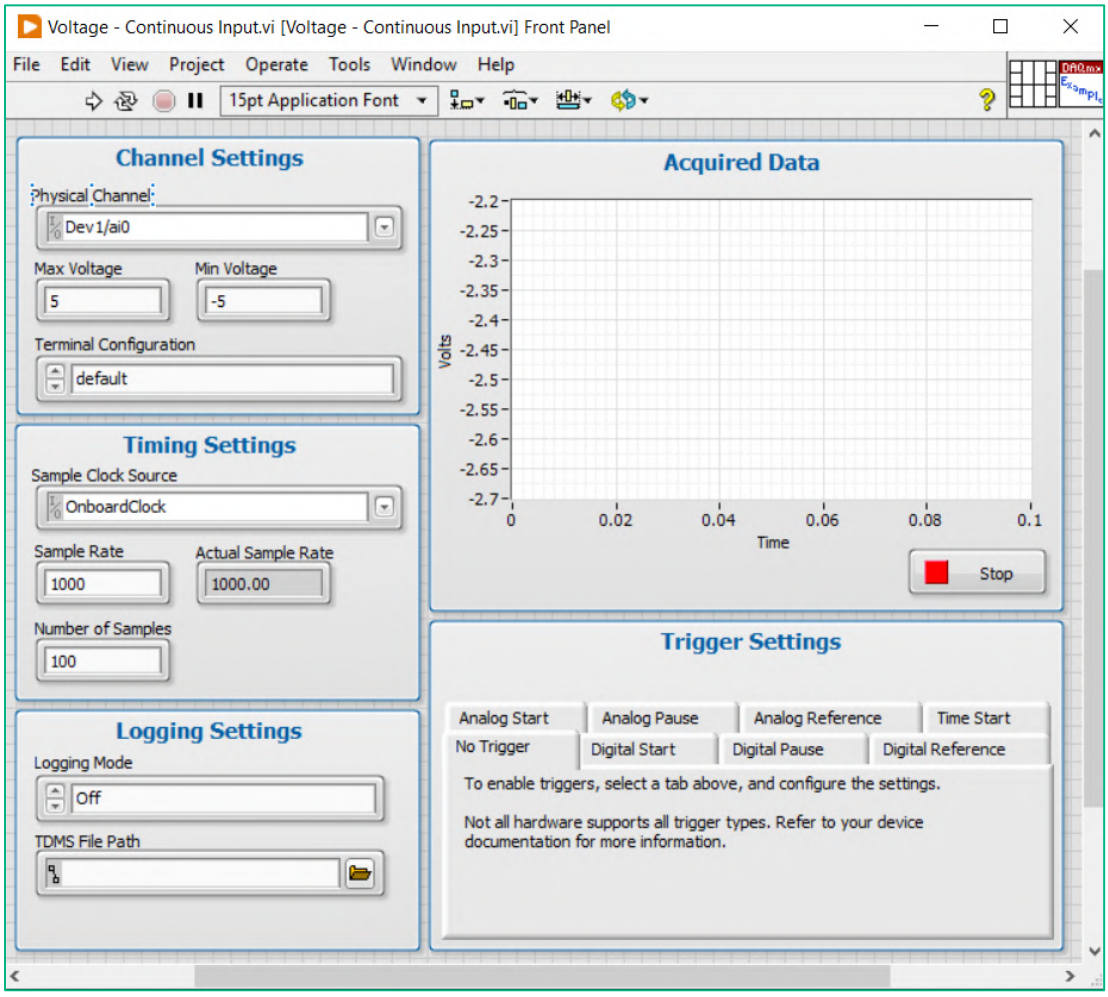


Figure 1. This Front Panel uses the Silver theme. Notice the different controls and indicators that are used and how they are organized.

3. To view the Block Diagram, select **Window » Show Block Diagram.**

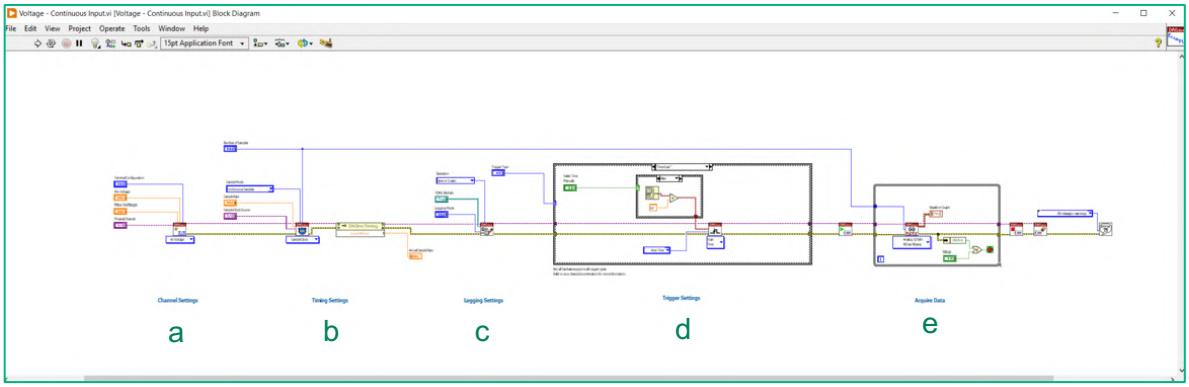


Figure 2. LabVIEW uses a graphical programming language and passes data via the wires that connect nodes.

4. This Block Diagram is organized into the following sections:

   a. **Channel Settings** – In this section, the **DAQmx Create Channel** configures the channel on the DAQ device that you intend to use. In this example, you can configure the range of the device and terminal configuration (single ended versus differential).

   b. **Timing Settings** – In the Timing Settings section, the **DAQmx Timing** is used to configure the sample clock on the DAQ device. Using this VI, you can configure the sample rate, the sample clock source, and the sampling mode you intend you use. In this example, the default settings use the on-board clock to sample continuously at 1000 Hz.

   c. **Logging Settings** – The **DAQmx Configure Logging** is used to set the location of the logged data file as well as the logging mode.

   d. **Trigger Settings** – In LabVIEW, you specify the triggering conditions that must be reached before acquisition begins. Once the conditions are met, the acquisition begins immediately.

   e. **Acquire Data** – In this section, the **DAQmx Read** is called inside a While Loop, allowing this code to continuously acquire data from the DAQ device until the Stop button is pressed.

5. Remember, for more information about the elements on your Block Diagram and Front Panel, press **<Ctrl+H>** to bring up the Context Help window. This can be an extra helpful tool while reviewing example code.

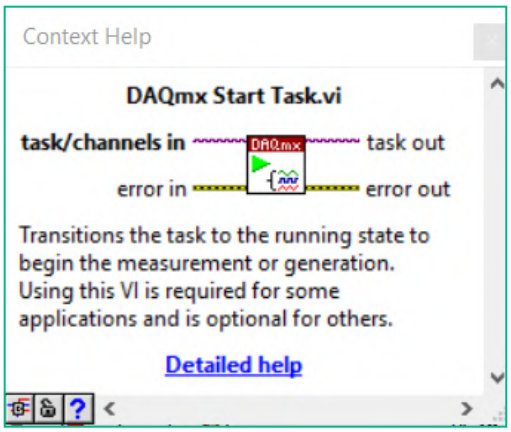


Figure 3. Context Help can provide a quick description of the elements in your Front Panel or Block Diagram.

6. On the Front Panel, select the channel to acquire data from. Click on the **Physical Channel** dropdown and select **Voltage_in/ai0**.

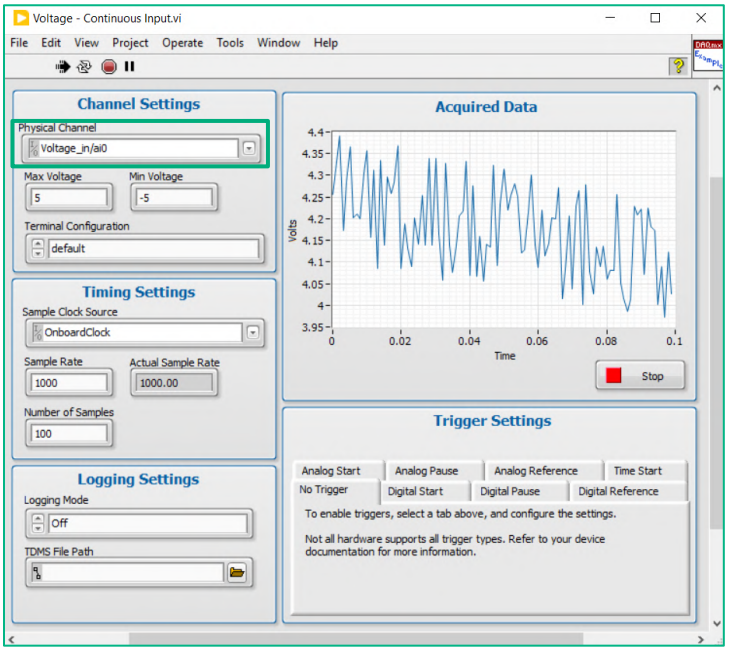


Figure 4. Use the controls on the Front Panel to select the channel, timing settings and logging settings.

7. Change the **Sample Rate** control to **10000** and set the **Number of Samples** to **1000**.

8. Click **Run** to start the VI. After some time, press the **Stop** button on the Front Panel.

<End of Exercise>

Bonus: Experiment With Other Example Programs (Optional)

Estimated time: 10 minutes.

Try exploring other example programs. Navigate to the example finder by going to **LabVIEW <Year> » Help » Example Finder**. Examples that might be useful to look at include:

1. For a temperature example: Navigate to the **Hardware Input and Output » DAQmx » Analog Input » Thermocouple – Continuous Input.vi**
    - Select **Temperature/ai0** from the Physical Channel control.

2. For a strain example: Navigate to the **Hardware Input and Output » DAQmx » Analog Input » Strain – Continuous Input.vi**
    - Select **Strain/ai0** from the Physical Channel control.
    - Select **Quarter Bridge I** from the Strain Configuration control.
    - Use **3.3** as the Voltage Excitation Value.

3. For a voltage example: Navigate to the **Hardware Input and Output » DAQmx » Analog Output » Voltage (non-regeneration) – Continuous Output.vi**
    - Select **Voltage_out/ao1** from the Physical Channel(s) control.
    - Select **1.00** from the Waveform Settings Frequency control.
    - Select **3.00** from the Waveform Settings Amplitude control.

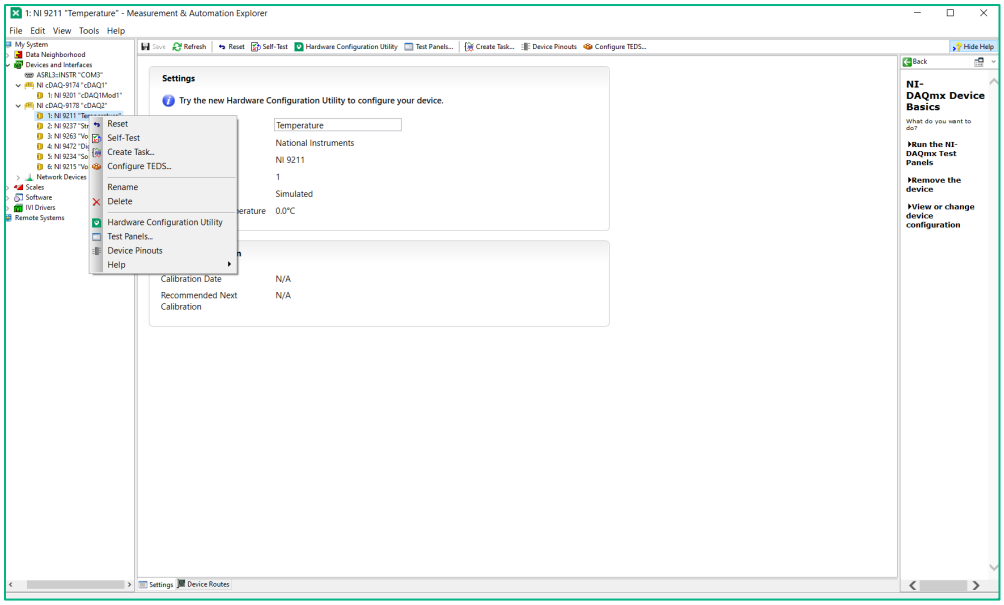


Figure 1. You can reset any hardware device to its default state in MAX.

<End of Exercise>